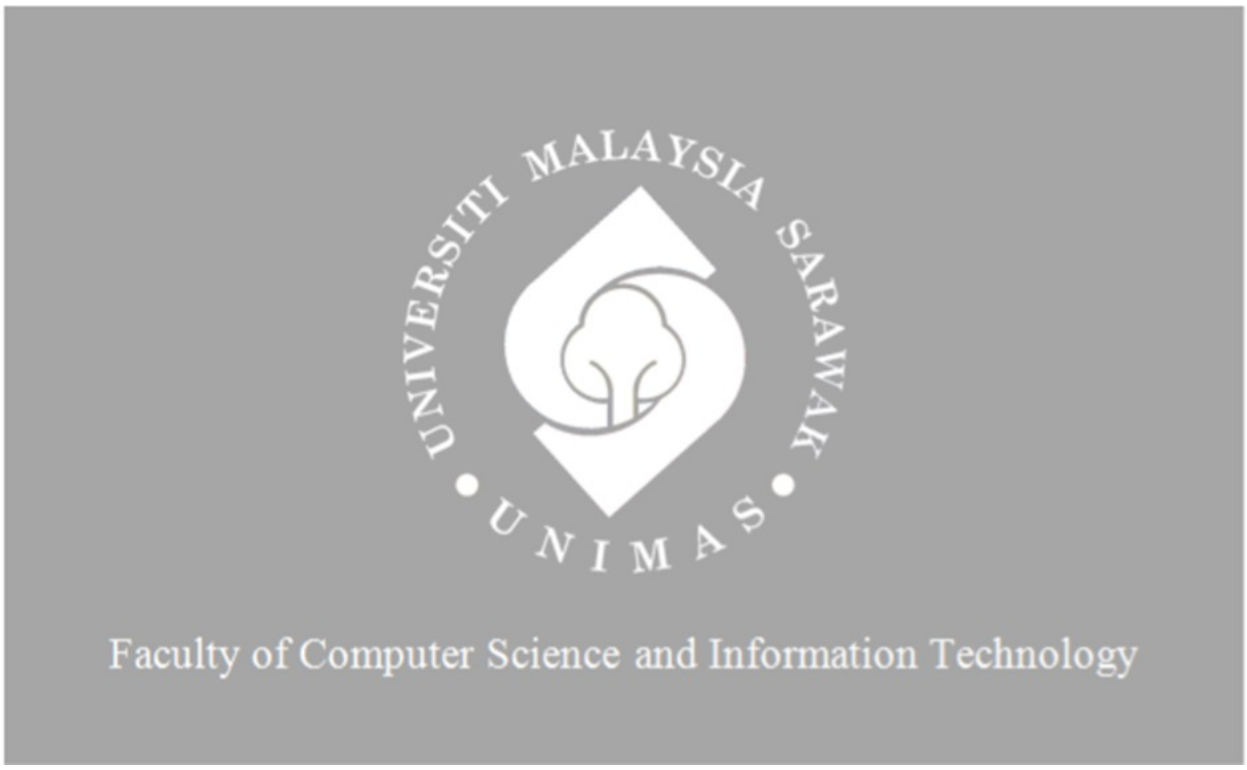



# *AI-Powered Software Bug Tracking Tool*

Ahmad Haizar bin Sahmat

Bachelor of Software Engineering with Honours

2024/2025

**UNIVERSITI MALAYSIA SARAWAK**

## THESIS STATUS ENDORSEMENT FORM

**TITLE** **AI-Powered Software Bug Tracking Tool**

**ACADEMIC SESSION: 2024/2025**

**AHMAD HAIZAR BIN SAHMAT**
**(CAPITAL LETTERS)**

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [ or for the purpose of interlibrary loan between HLI ]
5. ** Please tick ( √ )

| | | |
|---|---|---|
| ☐ | CONFIDENTIAL | (Contains classified information bounded by the OFFICIAL SECRETS ACT 1972) |
| ☐ | RESTRICTED | (Contains restricted information as dictated by the body or organization where the research was conducted) |
| ☑ | UNRESTRICTED | |

Validated by

| | |
|---|---|
| _____ | _____TPP_____ |
| (AUTHOR'S SIGNATURE) | (SUPERVISOR'S SIGNATURE) |
| Permanent Address | Dr. Tan Ping Ping |
| **SL 51, LORONG CAHAYA DAMAI 7B3D, TAMAN CAHYA PUTRI, 93050, KUCHING, SARAWAK** | |
| Date: 25th, Jul 2025 | Date: 25th, Jul 2025 |

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations

## DECLARATION

I hereby declare that the content of this project, titled "AI-Powered Software Bug Tracking Tool" is my original work, except for any research-based data or materials derived from other sources, which have been appropriately modified, cited, or referenced as required.

25th July 2025

(Ahmad Haizar bin Sahmat)
Faculty of Computer Science and Information Technology
Universiti Malaysia Sarawak

## ACKNOWLEDGEMENT

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Dr. Tan Ping Ping, for her invaluable guidance, continuous support, and encouragement throughout the course of this project. Her expertise, insightful feedback, and patience have been instrumental in shaping the direction and overall quality of this work. I would also like to extend my sincere thanks to Baitulmal Sarawak for generously providing the financial support that made this project possible.

## ABSTRACT

Most IT companies still rely on manual approaches to manage and track software bugs during testing. This traditional method is time-consuming, error-prone, and lacks efficiency particularly in large or complex software projects. To address these challenges, this project proposes the development of an AI-powered Software Bug Tracking Tool designed to streamline the process of reporting, tracking, categorizing, and prioritizing bugs. By leveraging Artificial Intelligence (AI) technologies, including machine learning, the system can automatically classify bugs based on their severity, providing valuable insights for better decision-making. Additionally, the system employs unsupervised learning techniques to detect patterns and relationships among bugs. An OpenAI GPT-4o model, accessed via the Azure OpenAI Service, is integrated into the system to automatically generate summaries of bug reports, helping teams quickly understand the key details. A user-friendly design is also provided to support collaboration between developers and testers, reducing the learning curve and improving overall software quality. The implementation of this system aims to enhance team productivity, ensure comprehensive bug tracking, and align with modern software development practices. Based on the usability testing with 10 participants, including software expert from various background such as software testers and developers, both participants and software expert found the duplicate detection and bug summarization features particularly useful, as they helped streamline the bug tracking process and improve understanding of reported issues. However, the bug severity prediction feature produced unexpected classifications in certain cases, indicating the need for additional training data and the potential integration of reinforcement learning to enable the model to continuously improve over time.

## ABSTRAK

Kebanyakan syarikat IT masih bergantung kepada kaedah manual dalam pengurusan dan penjejakan pepijat perisian semasa pengujian sistem. Kaedah tradisional ini bukan Sahaja memakan masa, malah, mudah terdedah kepada kesilapan, dan kurang cekap terutamanya dalam projek perisian yang besar atau kompleks. Bagi menangani masalah ini, projek ini mencadangkan pembangunan satu Alat Penjejak Pepijat Perisian berasaskan Kecerdasan Buatan (AI) yang direka untuk memudahkan proses menjejak, mengelaskan, dan memprioritikan pepijat. Dengan memanfaatkan teknologi AI termasuk pembelajaran mesin, sistem ini dapat mengklasifikasikan pepijat secara automatik berdasarkan tahap keterukan, sekali gus menyediakan pandangan bernilai untuk menyokong proses membuat keputusan. Selain itu, sistem ini turut menggunakan teknik pembelajaran tanpa seliaan, untuk mengesan corak dan hubungan antara pepijat. Sistem ini juga mempunyai reka bentuk yang mesra pengguna bagi menyokong kerjasama antara pembangun dan penguji, sekali gus mengurangkan masa latihan dan meningkatkan kualiti perisian. Pelaksanaan sistem ini bertujuan untuk meningkatkan produktiviti pasukan, memastikan penjejakan pepijat yang menyeluruh, dan sejajar dengan amalan pembangunan perisian moden.

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1:INTRODUCTION

## 1.1 Introduction

The testing phase plays a crucial stage in Software Development Lifecycle (SDLC), as it ensures the product is free of software bugs and meet all requirements before delivery. During this phase, software development and testing team collaborate to identify, analyze and resolve software bug within the system. When software testing is conducted efficiently, it leads to a reduction in expenses and time spent on rework or recalls, as issues are identified and resolved during the testing phase (Awork, 2024). Most tech companies nowadays rely on open-source bug tracking tools which available from certain provider to keep track of software bugs throughout testing. However, these existing tools often use traditional bug reporting approach, requiring testers and developers to report and analyze software bugs manually such as predicting severity, writing summaries of the bug, and identifying relationships between reported bugs. This manual approach can be time-consuming and prone to inconsistencies, which not only reduce the efficiency and accuracy of bug reporting and resolution but also leads to project time extensions. Therefore, by integrating AI in software bug tracking tool, development and testing teams can identify, analyze and resolve bugs more effectively, leading to improved productivity and software quality.

## 1.2 Problem Statement

Manual bug reporting is often time-consuming and prone to inefficiencies. Some IT companies are still relying on Microsoft Excel and traditional open-source bug tracking tools, which lack features to ease reporting, tracking and analyzing software bugs, making these processes challenging. Inconsistencies in categorizing and assessing the severity of bugs during bug reporting could cause not only slows down debugging process but can also lead to inaccurate bug prioritization during software testing. Other than that, developers frequently struggle to identify related bugs across different modules, missing valuable insights into potential root causes. Hence, an AI-powered solution could address these challenges by improving bug reporting and tracking experience, ensuring consistent bug categorization, and automatically finds related bugs for improved analysis.

## 1.3 Objectives

The objectives of the project are as the following:

I. To develop a bug tracking tool that able to keep track of software bugs during testing phase.

II. To accelerate bug reporting process through the integration of Artificial Intelligence (AI) technologies.

III. To integrate Machine Learning (ML) subset of AI that learns bugs pattern and predict severity.

IV. To evaluate the effectiveness of the proposed bug tracking tool for testers and developers during testing.

## 1.4 Methodology

This project will be implemented using one of the methodologies that leveraging object-oriented principles within the Unified Process framework, which is Unified Process methodology (UP). UP is a software development methodology that follows an iterative and incremental approach to develop systems (Lethbridge & Laganière, 2004, p. 432). It divides the development lifecycle into four phases: Inception, Elaboration, Construction, and Transition, each with specific objectives and milestones. Figure 1 shows the Unified process methodology and its phases.

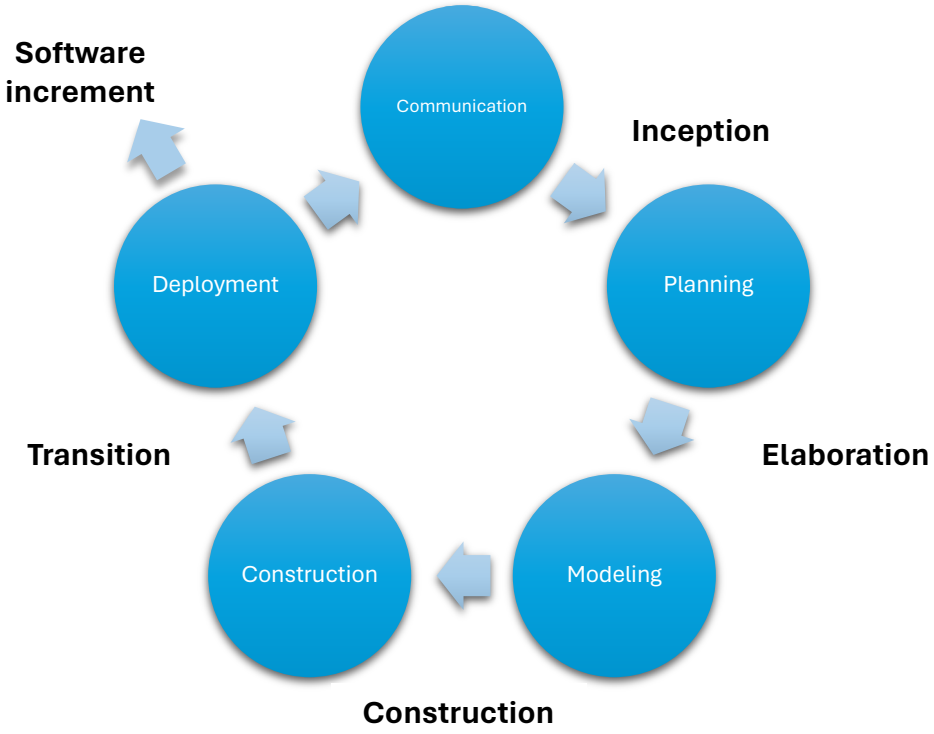



Figure 1: Unified Process Methodology

### 1.4.1 Inception

Define the project's vision, scope, and feasibility, defining the initial requirements, identify risks, and create an initial plan. The goal is to understand the project’s purpose and lay the groundwork for the next phases.

### 1.4.2 Elaboration

Refine requirements, system architecture, address risks, and develop system prototypes for complex or high-risk components. This phase solidifies the project's foundation and prepares for development.

### 1.4.3 Construction

Build the system incrementally through iterations, implementing and testing features, and integrating components. The focus is on developing a functional system, collecting feedback, and ensuring software quality.

### 1.4.4 Transition

Finalize the product, perform testing, fix issues, and deploy it to the production environment. This phase ensures the system is stable, delivers the final product to users, and provides necessary training and documentation.

## 1.5 Scope

The AI-powered software bug tracking tool aims to make software testing more efficient in tracking, identifying and reporting bugs. By utilizing Artificial Intelligence (AI) technologies, the tool will be able to predict bug severity, summarize bug details and analyze the relationships between bugs more effectively. This tool will reduce the manual effort needed, allowing testers to expand the test coverage and work efficiently. Automating bug analysis and reporting will help the tool to speed up testing phase and make developers and testers more productive.

## 1.6 Significance of Project

The significances of this project are:

I. The software bug tracking tool able to accelerate bug reporting and tracking processes through the integration of Artificial Intelligence (AI) technologies.

II. The software bug tracking tool able to help Software Development Team in terms of productivity.

III. Increase the effectiveness of testers and developers during testing process.

## 1.7 Project Schedule

Figure 2 and Figure 3 present the Gantt charts outlining the project timeline for Semester 1 and Semester 2 respectively."

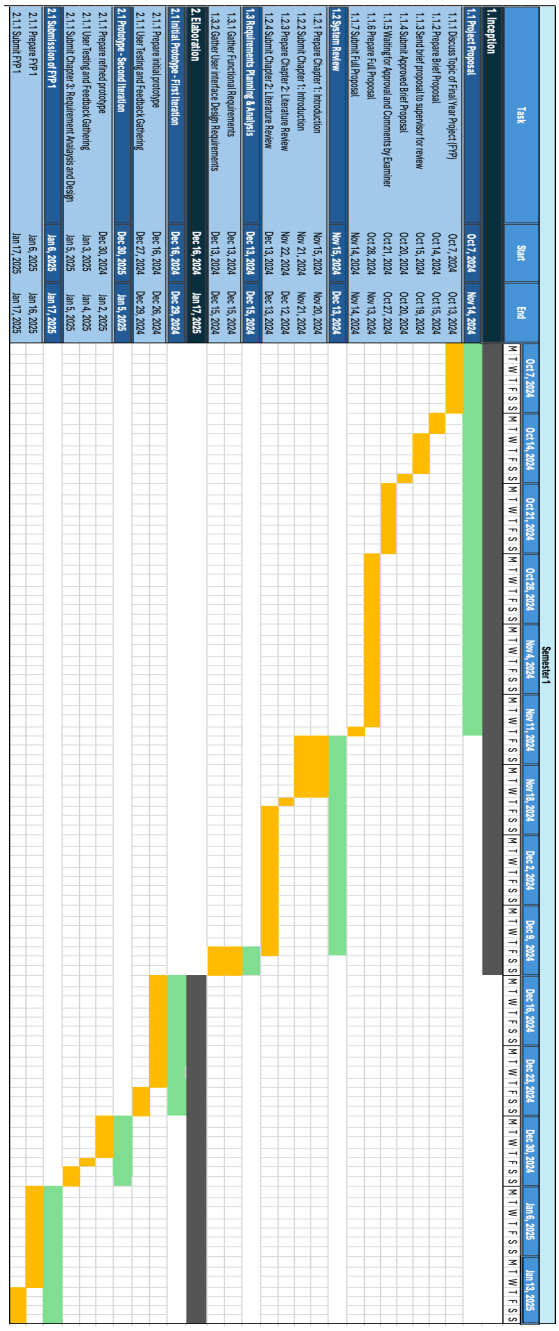



Figure 2: Gantt chart for semester 1

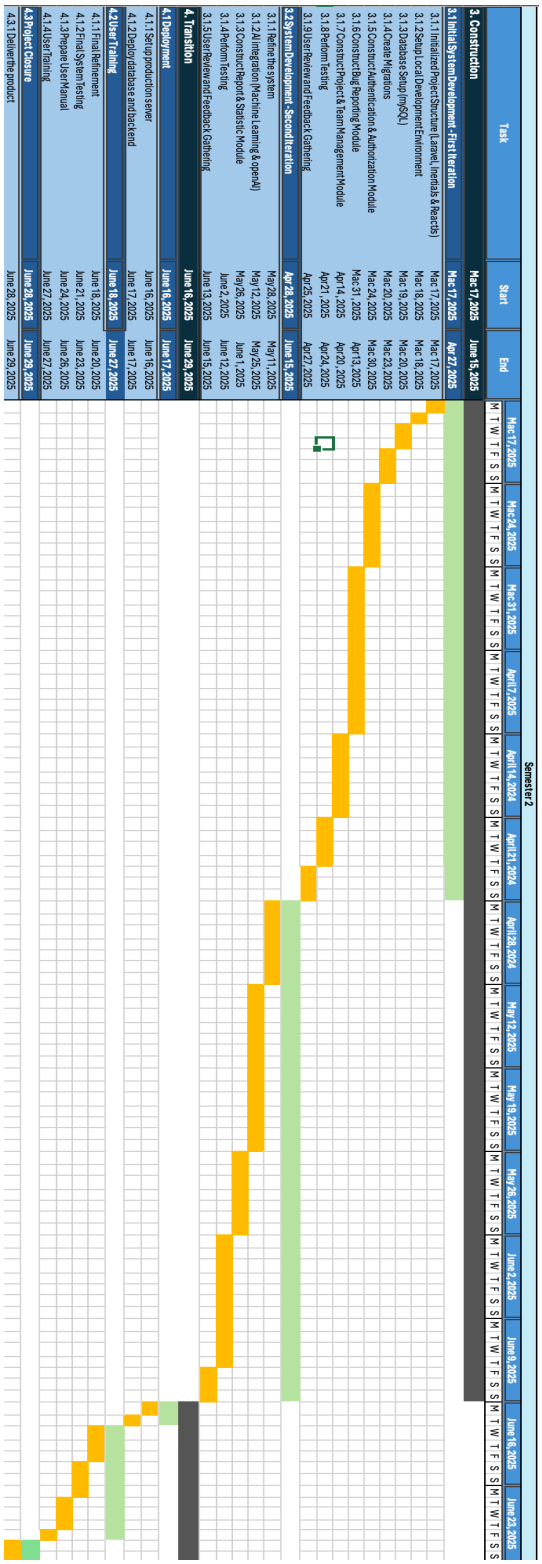



Figure 3: Gantt chart for semester 2

## 1.8 Expected Outcome

The expected outcome of this project is a fully functional AI-Powered Software Bug Tracking Tool that able to speed up software bug management and improve productivity for both software testers and developers. The integration of AI will make bug reporting, tracking and analyzing process faster and more accurate, helping teams to work efficiently.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Introduction

This chapter includes a review of existing software bug tracking tools along with the proposed system and software technologies during development. The purpose of the literature review in this project is to provide a clear understanding of the current knowledge related to the project's topic, facilitating the development of the proposed system. It examines existing software bug tracking tools, identifies relevant software technologies, and highlights the challenges and opportunities in the field. By understanding these aspects, the review aims to inform the design and implementation of the proposed system, ensuring it meets current industry standards and addresses the existing gaps. Additionally, there has been some research carried out to provide a clearer understanding of the proposed project.

## 2.2 Review on existing tools

Reviewing existing tools is part of the literature review process. It involves studying current solutions to identify their strengths, weaknesses, and overall effectiveness. This analysis provides valuable insights into what features are commonly implemented, what limitations users face, and where improvements can be made.

### 2.2.1 Microsoft Excel

Excel is widely used within organization as it easy to manage and track sets of data. Nowadays, having skill in Microsoft Excel is crucial as most of the employee from different team will be dealt with excel. The high usage of excel in IT field is to close the gap between technical

and non-technical teams. Figure 4 shows the interface of Microsoft Excel and Table 1 below shows the features and disadvantages of Microsoft Excel as bug tracking tool.

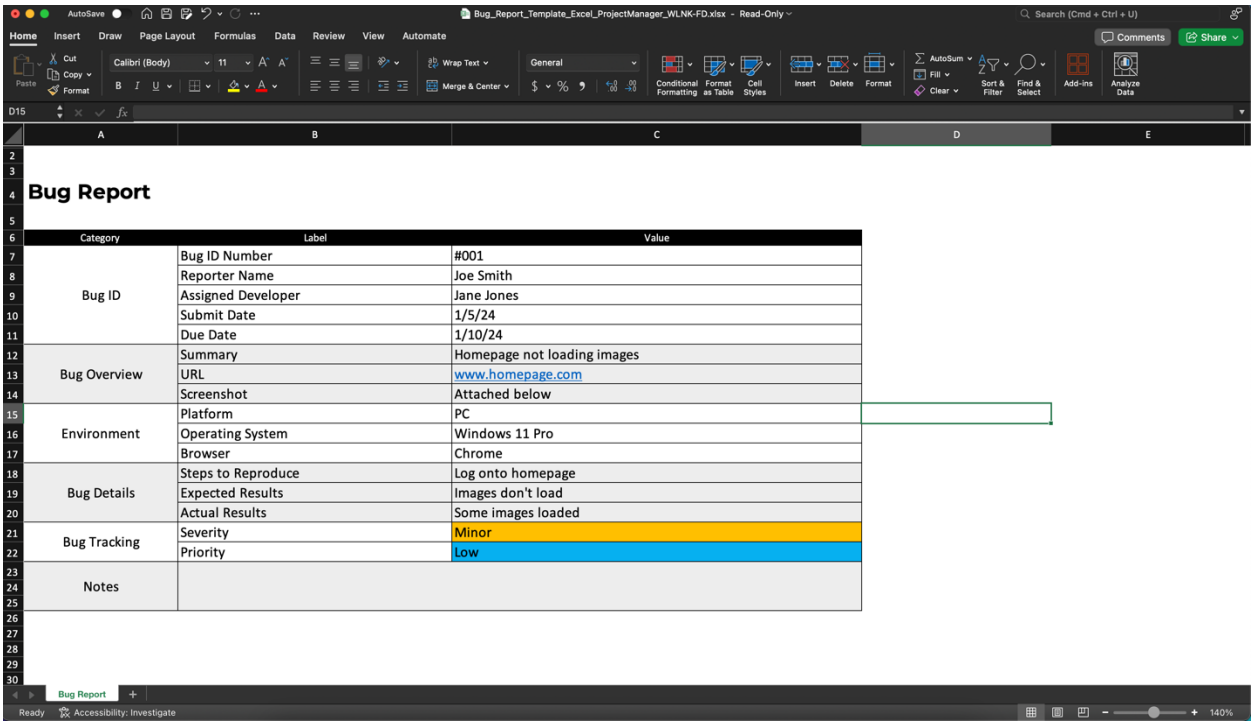



Figure 4: Using Excel as Bug Tracking Tool

Table 1: Features and disadvantages of Excel as Bug Tracking Tool

| Features | Review |
|---|---|
| User Interface | • Widely recognized for its simplicity and familiarity<br>• The user interface is not suitable for software bug management |
| Notification | • No built-in notification system for bug tracking |
| Bug Management | • Manual Entry for bug id, reporter, assignee, severity, priority and date of issue |

| | |
|---|---|
| Bug Analysis | • Relationship between bugs need to be determine manually |
| Collaboration | • Poor collaboration |
| AI integration | • AI integration is included but limited |
| Methodology Compatibility | • Agile processes must be implemented manually, requiring users to manage tasks<br>• Waterfall |

### 2.2.2 Mantis Bug Tracker

Mantis Bug Tracker is an open-source bug tracking and project management tool designed to help teams efficiently report, track, and manage software bugs. It is widely used by development teams for tracking bugs, managing tasks, and ensuring smooth communication within teams. Table 2 below shows the features and disadvantages of Mantis Bug Tracker. Figure 5 shows the interface of Mantis Bug Tracker.

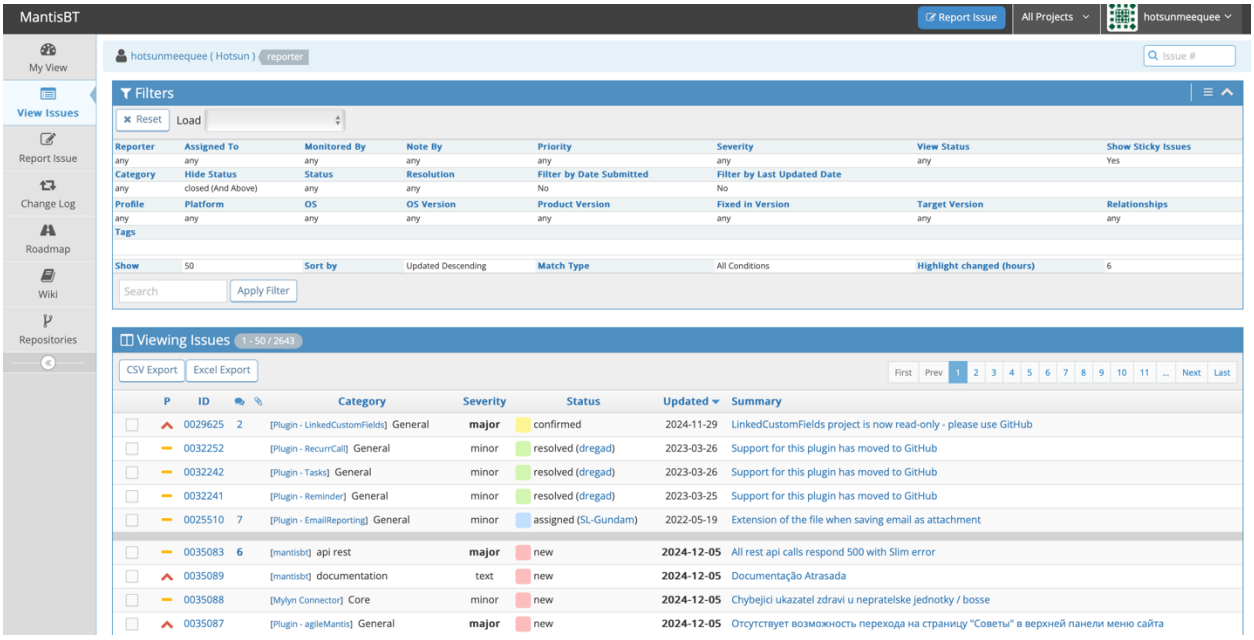



Figure 5: Mantis Bug Tracker

Table 2: Features and disadvantages of Mantis

| Features | Review |
|---|---|
| User Interface | • Outdated User Interface |
| Notification | • Built-in notification system for bug tracking is provided |
| Bug Reporting | • Bug id, reporter, and date of issue automatically defined |

| | |
|---|---|
| | • Manual entry for assignee, severity and priority |
| Bug Analysis | • Relationship between bugs need to be determined manually |
| Collaboration | • Support real-time collaboration |
| AI integration | • AI integration is not included |
| Methodology Compatibility | • Waterfall, Agile, Scrum etc. |

### 2.2.3 Jira

Jira is a popular, comprehensive project management and issue tracking software developed by Atlassian. It is widely used for managing bug tracking and software development projects. Jira has integrated some AI-powered features to streamline the bug tracking process, enhance productivity, and reduce manual effort. Figure 6 shows the interface of Jira and Table 3 below shows the features and disadvantages of Jira as a software bug tracking tool.

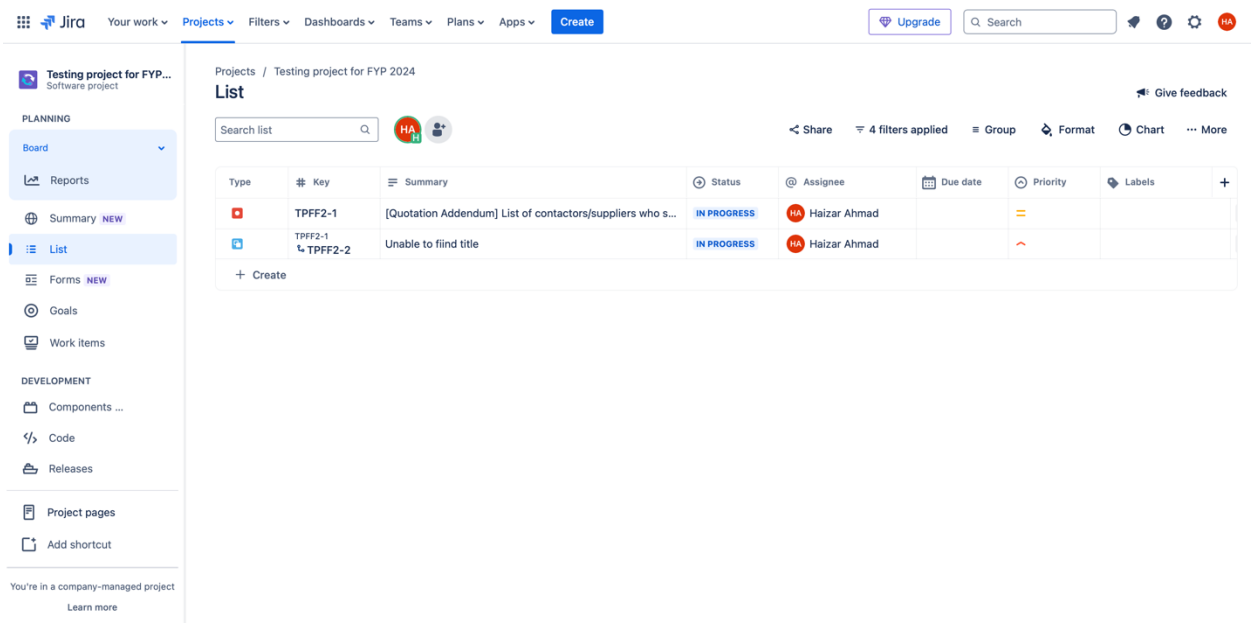



Figure 6: Jira Bug Tracking Tool

Table 3: Features and disadvantages of Jira

| Features | Review |
|---|---|
| User Interface | • Support customizability<br>• Modern but complex User Interface |
| Notification | • Built-in notification system for bug tracking is provided |
| Bug Reporting | • Bug ID, reporter, and date of issue automatically defined<br>• Manual entry for bug summary, assignee, severity, and priority<br>• Bug description can be generated and summarized with AI |
| Bug Analysis | • Relationship between software bugs need to be determined manually<br>• Bug discussion can be summarized with AI |
| Collaboration | • Support real-time collaboration |
| AI integration | • AI integration is included |
| Methodology Compatibility | • Waterfall, Agile, Scrum etc. |

### 2.2.4 ClickUp

ClickUp is a project management and productivity tool designed to help teams manage tasks, projects, and workflows. While it is primarily a project management tool, it provides functionalities that can be adapted for bug tracking and analysis. ClickUp provides features that are beneficial for managing software bugs and tracking issues within a project. Figure 7 shows the interface of Clickup. Table 4 below shows the features and disadvantages of ClickUp as a bug tracking tool.

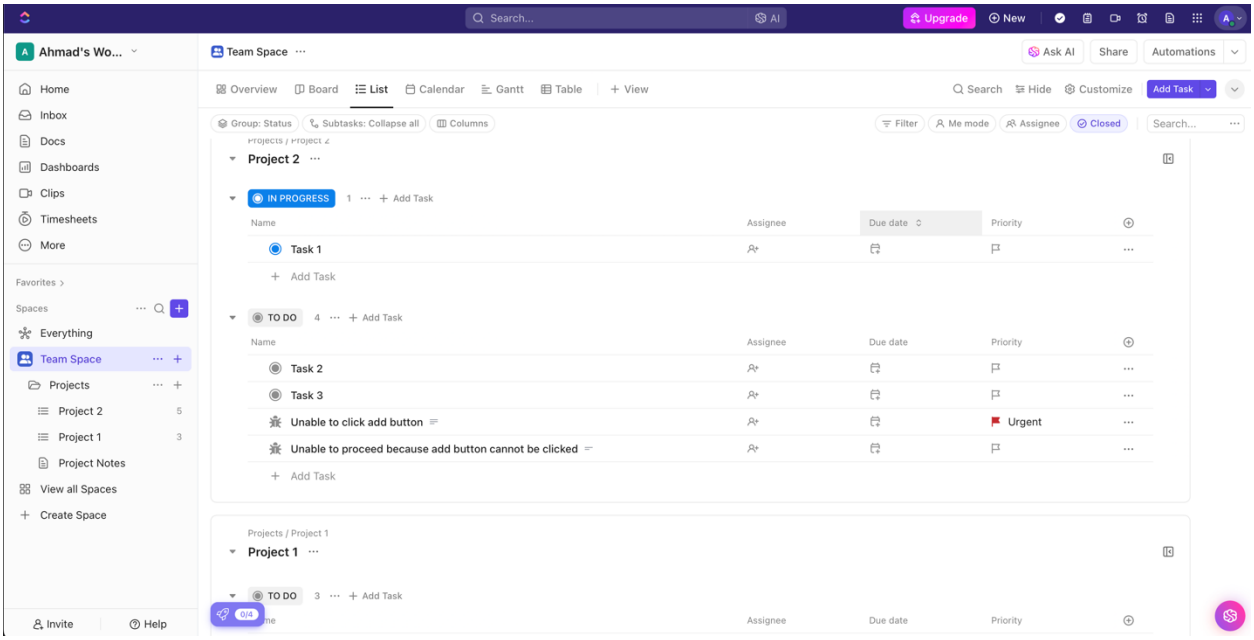



Figure 7: Clickup Bug Tracking Tool

Table 4: Features of ClickUp

| Features | Review |
|---|---|
| User Interface | • Support customizability<br>• Modern User Interface |
| Notification | • Built-in notification system for bug tracking is provided |
| Bug Reporting | • Bug ID, reporter, and date of issue automatically defined<br>• Manual entry for bug summary, assignee, and priority<br>• Bug severity information is not included<br>• Bug description can be generated and summarized with AI |
| Bug Analysis | • Relationship between software bugs can be determined with the help of AI<br>• Bug discussion can be summarized with AI |
| Collaboration | • Support real-time collaboration |
| AI integration | • AI integration is included |
| Methodology Compatibility | • Waterfall, Agile, Scrum etc. |

## 2.3 Comparison of existing tools and the proposed tool

The existing tools that have been reviewed, such as Microsoft Excel, Mantis Bug Tracker, ClickUp, and Jira each has its own strengths and weaknesses. Table 5 below shows the comparison of existing tools and the proposed tool.

Table 5: Comparison of existing tool and proposed system

| **Criteria** | **Excel** | **Mantis** | **Jira** | **ClickUp** | **Proposed System** |
|---|---|---|---|---|---|
| **User Interface** | Widely recognized for its simplicity and familiarity | Outdated User Interface | Modern design but with complex user interface | Modern User Interface design | Straightforward and modern user interface |
| **Notification** | No | Built-in notification system is provided | | | |
| **Bug Reporting** | Bug reporting relies heavily on manual entry | Some of the bug details relies on manual entry | Some of the bug details are automatically generated | Requires less manual entry as it is mostly automated but does not include bug severity information | Require less manual entry as bug reporting is mostly automated |

| | | | | |
|---|---|---|---|---|
| **Bug Analysis** | No | Able to analyze software bugs, but user intervention is required | AI chatbot is provided to assist in conducting bug analysis | Automatically conduct bug analysis with the help of AI |
| **Collaboration** | Poor | Support real-time collaboration | | |
| **AI Integration** | No | Yes | | |
| **Methodology Compatibility** | Waterfall | Supports common development methodologies, including Agile, Waterfall, Scrum etc. | | |

## 2.4 Summary comparison of existing tools and the proposed tool

The existing tools that have been reviewed, such as Microsoft Excel, Mantis Bug Tracker, ClickUp, and Jira, each have their own strengths and weaknesses. While Excel offers simplicity, it lacks automation and scalability. Mantis provides basic bug tracking features but has limited AI support. ClickUp and Jira are more comprehensive, but they can be complex to configure and may not be optimized for software bug management.

## 2.5 Brief Overview of the proposed system

The AI-Powered software bug tracking tool will addresses many of limitations found in the existing bug tracking tools. The proposed system will enhance the efficiency and accuracy of software bug management. The integration of AI technologies will accelerate bug reporting and provide helpful analysis, reducing the manual effort required by software testers and development

teams, thereby improving overall team productivity. It offers a modern and straightforward interface that simplifies the navigation for both technical and non-technical users, ensuring seamless adoption across teams. Real-time collaboration and built-in notifications features are also included to keep team members informed. Lastly, the tool will support common development methodologies like Agile, Waterfall and Scrum.

## 2.6 Review on Software Technologies

During the development of the proposed system, Microsoft Visual Studio Code and PyCharm will be used as the main tools for writing and editing code. For the front-end, the User Interface (UI) will be built using React JS to create a sleek, modern, and user-friendly design. On the back-end, the system will be develop using Laravel, which is a popular PHP framework that helps organize code and makes the system easier to maintain. To connect React JS and Laravel without the need to develop an API, Inertia.js will be used as a bridge, enabling smooth communication between the front end and back end while maintaining a single-page application experience. MySQL will be used to manage and organize data efficiently. Other than that, Microsoft Azure's Cloud Services will be used to include the AI features to the bug tracking tool. The system also will be using Flask, a lightweight Python framework, to integrate the Machine Learning model into the system for users to access.

For the machine learning model part, several studies highlight the importance of transforming textual data into numerical form before classification process to enable machine learning models to process and analyze it effectively (Otoom et al., 2016). This transformation typically involves tokenization and vectorization, as computers cannot directly interpret raw text.

In this project, among various vectorization techniques, Term Frequency-Inverse Document Frequency (TF-IDF) has been chosen due to its ability to represent text data by identifying the importance of individual words within a given dataset. Once tokenized and vectorized, the processed input will be fed into the machine learning models to do the prediction. According to research conducted by Hamouri et al. (2023) on predicting bug severity using a machine learning approach, the results showed that among six different machine learning algorithms tested, Multinomial Logistic Regression achieved a decent accuracy score, averaging 0.70 on six datasets from open-source projects. Therefore, Multinomial Logistic Regression algorithm will be selected as the preferred algorithm for bug severity classification in this project.

## 2.7 Summary

After reviewing the literature, it is shown that existing bug tracking tools, such as Microsoft Excel, Mantis Bug Tracker, ClickUp and Jira, have contributed to improve software bug tracking and management. These tools provide common features which have simplified the bug tracking processes. Most existing tools lack automation and AI integration, which limits their effectiveness in enhancing bug reporting and analysis. To address these limitations, there is a need for a tool that incorporates AI technologies to improve both bug reporting, tracking and analysis processes. Hence, this tool should offer a promising solution by addressing the limitations identified during the review on existing tools. Developing such a solution will bridge the gaps identified and improve the overall efficiency of bug tracking processes.

# CHAPTER 3: REQUIREMENT ANALYSIS AND DESIGN

## 3.1 Introduction

Requirement analysis and design are critical stages in software development that bridge user needs and technical solutions. Requirement analysis focuses on gathering, understanding, and documenting what the system must do to meet user expectations. This includes identifying functional requirements, non-functional requirements, and system interface design translates these requirements into a blueprint for building the system. It involves creating architectural diagrams, database design, and detailed specifications for components and their interactions. These stages ensure the system is well-defined, feasible, and aligned with client goals before implementation begins.

## 3.2 Methodology

The Unified Approach methodology was chosen to guide the elicitation of requirements and the development of the system design. Its structured framework ensured that all functional and non-functional requirements were accurately captured and aligned with stakeholder expectations. The planning process involved two key phases: Inception and Elaboration. During the Inception Phase, interviews were held with the client to gain valuable insights into their needs, helping to define the initial set of requirements. During the Elaboration Phase, these requirements were refined and expanded to create a more detailed system design.

### 3.2.1 Inception

In the Inception Phase, interview was conducted with the user from Tun Abang Haji Openg Digital Centre (TAHODC) to identify and analyze the functional and non-functional requirements of the project. The main purpose of this interview is to gather insights from experienced software development teams to identify the requirements that should be included in the software bug tracking tool. Before the interview, a pre-survey was performed to assess the user's understanding and knowledge of software bug tracking tools. The survey also aimed to study the organization's workflow and practices in their software development process and to clarify any potential ambiguities that might arise during the interviews.


Figure 8: Results of bug tracking tool existence

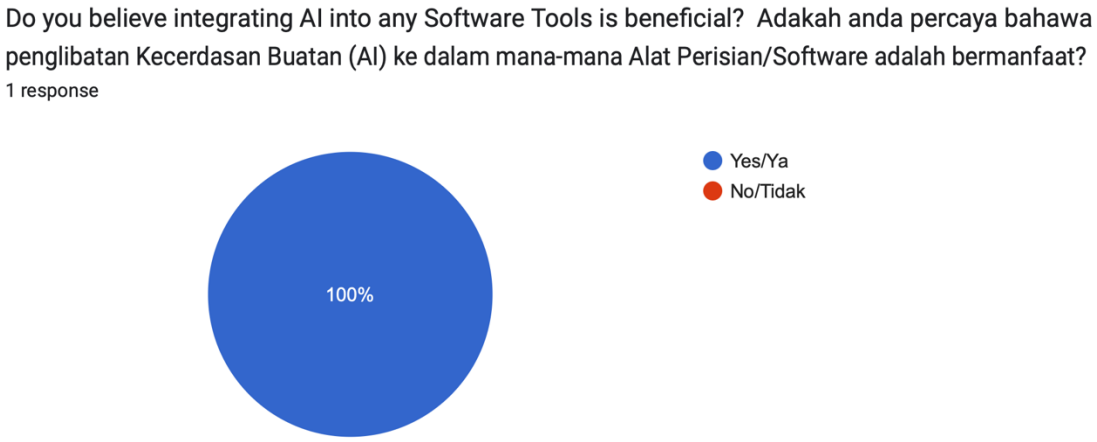



Figure 9: Results of bug tracking tool usage

Based on the results of the figure 8 and figure 9, it shows that the organization is aware of software testing process but does not currently practice using software bug tracking tool to manage bugs discovered during software development and they also believed that integrating AI into the system could bring significant benefits in software development. During the interview, the initial requirements obtained from analyzing the existing tools were revised based on user feedback, and any additional requirements from the user were added to ensure the system aligns with current market demands and user expectations.

### 3.2.2 Elaboration

In the Elaboration Phase, the initial user interface design was developed based on the gathered requirements. The early design was then presented to user to gather feedback, which helped identify areas for improvement. This feedback was used to refine the interface to enhance usability, accessibility, and modern design standards, ensuring a user-friendly experience before proceeding with detailed system development. Any additional requirements or changes requested

by user were added and finalized to ensure the system aligns with user expectations before construction phase.

### 3.3 Requirement Analysis

In this section, the Use Case Diagram, functional and non-functional requirements of the system are outlined. Use Case Diagrams provide a high-level view of the system's functionality from the perspective of end users. This is important in the Unified Approach as it allows to clearly define the system's functional requirements early in the process. Functional requirements specify the core features and behaviors that the system must support, such as user interactions, data processing, and business logic. Non-functional requirements, on the other hand, define the system’s performance attributes, such as reliability, security, scalability, and usability. Both sets of requirements are essential for ensuring that the system not only meets its functional objectives but also performs effectively under various conditions, delivering a seamless user experience.

### 3.3.1 Use Case Diagram


Figure 10: Use Case Diagram

The figure 10 illustrates the interactions between a software tester, software developer, administrator and the system to achieve specific goals. It outlines each actor's actions and their relationship with the system. The Use Case Diagram will guide development and testing, ensuring the system meets its objectives, and provide valuable documentation for future modifications or maintenance.

### 3.3.2 Use Case description

Tables below are the description of each use case, including its actor, pre and post condition, main path and exception.

Table 6: Use case description for report bug

| | |
|---|---|
| **Use Case** | Manage Bug |
| **Use Case Description** | This use case to allow Software Tester and Developer to manage bug |
| **Actor/s** | 1. Software Tester<br>2. Software Developer |
| **Pre-Condition** | 1. Project exists |
| **Post-Condition** | 1. The system will notify assigned developer |
| **Main Path** | 1. Users go to issue module |
| **Exception** | 1. System alerts user to add issues if there is no any |

Table 7: Use case description for predict bug severity

| | |
|---|---|
| **Use Case** | Predict Bug Severity |
| **Use Case Description** | This use case to allow Software Tester to predict the severity of an issue based on title using AI Technologies |
| **Actor/s** | Software Tester |
| **Pre-Condition** | 1. Title of the bug is filled |
| **Post-Condition** | 2. The system will predict the severity and displays confirmation message |

| | |
|---|---|
| **Main Path** | 1. Software Tester click add bug button<br>2. Software Tester fills in bug title<br>3. System will automatically predict the severity of the bug based on its title<br>4. The system displays confirmation message |
| **Exception** | 1. System alerts user to fill in required fields |

Table 8: Use case description for enhance bug description

| | |
|---|---|
| **Use Case** | Summarize bug details |
| **Use Case Description** | This use case to allow Software Tester to summarize the bug using OpenAI Technologies |
| **Actor/s** | 1. Software Tester<br>2. Software Developer |
| **Pre-Condition** | 1. Summary of the bug is filled |
| **Post-Condition** | 1. The system will summarize the description of bug<br>2. The system will display the confirmation message |
| **Main Path** | 1. Software Tester click add bug button<br>2. Software Tester fills in bug title<br>3. System will automatically predict the severity of the bug based on its title<br>4. The system displays confirmation message |
| **Exception** | 1. System alerts user to fill in required fields |

Table 9: Use case description for notify developer

| **Use Case** | Notify Developer |
|---|---|
| **Use Case Description** | This use case to allow the system to notify software developer upon bug status update |
| **Actor/s** | Software Tester |
| **Pre-Condition** | 1. Software Tester assigns bug to software developer<br>2. Software developer exists |
| **Post-Condition** | 1. The system will notify software developer regarding assigned bug |
| **Main Path** | 1. Software Tester click add bug button<br>2. Software Tester fills in bug details<br>3. Software Tester assigns a bug to desired software developer<br>4. Software Tester click submit button<br>5. The system notifies software developer regarding assigned bug |
| **Exception** | 1. System alerts user to fill in required fields |

Table 10: Use case description for notify tester

| **Use Case** | Notify Tester |
|---|---|
| **Use Case Description** | This use case to allow the system to notify software tester upon bug status update |
| **Actor/s** | Software Developer |

| | |
|---|---|
| **Pre-Condition** | 1. Software Tester assigns bug to software developer<br>2. Software Tester exists |
| **Post-Condition** | 1. The system will notify software tester regarding assigned bug |
| **Main Path** | 1. Software Tester click assigned bug<br>2. Software Tester updates the bug status<br>3. The system notifies software tester when bug status changes |
| **Exception** | N/A |

Table 11: Use case description for update bug status

| | |
|---|---|
| **Use Case** | Update Bug Status |
| **Use Case Description** | This use case to allow the software developer to update the bug status |
| **Actor/s** | Software Developer |
| **Pre-Condition** | 1. Software Tester assigns bug to software developer |
| **Post-Condition** | 1. The system will notify software tester when the bug status changes<br>2. The bug status changed |
| **Main Path** | 1. Software developer click assigned bug<br>2. Software developer add comments on the issue<br>3. Software Tester updates the bug status<br>4. The bug status changes |

| | |
|---|---|
| **Exception** | N/A |

Table 12: Use case description for add comment

| | |
|---|---|
| **Use Case** | Add comment |
| **Use Case Description** | This use case to allow the software developer and software developer to comment on bug |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Software Tester create a bug and assigns it to software developer |
| **Post-Condition** | 1. The system will notify software tester and software when comment is added |
| **Main Path** | 1. Software developer/Software Tester click on desired bug<br>2. Software developer/Software Tester add comments under activities section<br>3. The system will display all comments regarding the issue<br>4. The system will notify software tester and software when comment is added |
| **Exception** | N/A |

Table 13: Use case description for view all bugs

| | |
|---|---|
| **Use Case** | View all bugs |
| **Use Case Description** | This use case to allow the system to show all bugs from the project |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Project exists<br>2. Project have logged bugs |
| **Post-Condition** | 1. The system will show all reported bug |
| **Main Path** | 1. Software Tester/Software Developer click issues module<br>2. The system shows all reported bugs with its details |
| **Exception** | 1. System shows no bug is yet reported in the project |

Table 14: User case description for bug similarity detection

| | |
|---|---|
| **Use Case** | Bug similarity detection |
| **Use Case Description** | This use case to allow the system to find bug similarity using Machine learning technique |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Project exists<br>2. Project have logged bugs |
| **Post-Condition** | 1. The system will show all similar reported bug |
| **Main Path** | 1. Software Tester/Software Developer click issues module |

| | |
|---|---|
| | 2. The system shows all reported bugs with its details<br>3. View any issue<br>4. All similar issue will be shown |
| **Exception** | 1. System shows no bug is yet reported in the project |

Table 15: Use case description for create project

| | |
|---|---|
| **Use Case** | Create Project |
| **Use Case Description** | This use case to allow the software developer and software tester to create a new project |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Team has created |
| **Post-Condition** | 2. New project created |
| **Main Path** | 1. Software Tester/Software Developer navigate to project module<br>2. Software Tester/Software Developer click on the create project button<br>3. Fill in new project details<br>4. Select Project team<br>5. Click on create project button |
| **Exception** | 1. System alerts user to fill in required fields |

Table 16: Use case description for manage project

| **Use Case** | Manage Project |
|---|---|
| **Use Case Description** | This use case to allow the software developer and software tester to manage project |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Project exists |
| **Post-Condition** | 1. System allow user to update, delete and view project |
| **Main Path** | 1. Software Tester/Software Developer navigate to project module<br>2. Software Tester/Software Developer click on existing project<br>3. System allow user to update, delete and view project |
| **Exception** | 1. System alerts user to fill in required fields |

Table 17: Use case description for manage team

| **Use Case** | Manage Team |
|---|---|
| **Use Case Description** | This use case to allow the software developer and software tester to manage team |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Project exists<br>2. Team with team members exist |
| **Post-Condition** | 1. System allow user to update, delete and view Team |
| **Main Path** | 1. Software Tester/Software Developer navigate to Team module<br>2. Software Tester/Software Developer click on existing Team<br>3. System allow user to update, delete and view team |
| **Exception** | 2. System alerts user to fill in required fields |

Table 18: Use case description for manage sprint cycle

| **Use Case** | Manage Sprint Cycle |
|---|---|
| **Use Case Description** | This use case to allow the team to manage sprint cycle of the project |
| **Actor/s** | 1. Software Developer<br>2. Software Tester |
| **Pre-Condition** | 1. Project exists |

| | |
|---|---|
| | 2. Team with team members exist<br>3. Project has sprint cycles |
| **Post-Condition** | 1. System allow user to update, delete and view sprint cycle of a project |
| **Main Path** | 1. Software Tester/Software Developer navigate to Project Sprint module<br>2. Software Tester/Software Developer click on existing sprint<br>3. System allow user to update, delete and view desired sprint |
| **Exception** | 1. System alerts user to fill in required fields |

Table 19: Use case description for view project summary

| | |
|---|---|
| **Use Case** | View Project Summary |
| **Use Case Description** | This use case to allow the team to view the summary of a project |
| **Actor/s** | 1. Software Tester<br>2. Software Developer |
| **Pre-Condition** | 1. Project exists<br>2. The project has reported bugs |
| **Post-Condition** | 1. System will display the summary of the project such as total of open issue, severity of bugs etc. |
| **Main Path** | 1. Software Tester/Software Developer navigate to desired project |

| | |
|---|---|
| | 2. Software Tester/Software Developer select statistic module<br>3. System will display the summary of the project such as total of open issue, severity of bugs etc. |
| **Exception** | N/A |

### 3.3.3 Functional requirements

The functional requirements for the system are categorized into several key areas to ensure comprehensive coverage of all essential features. These categories include authentication, authorization levels, user profile management, project management and bug reporting, project summary and AI and Machine Learning Integration.

#### 3.3.3.1 Authentication and Authorization

Table 20: Functional requirements for Authentication

| Req. No. | Requirement Description |
|---|---|
| **REQ-1** | User shall be able to login and redirect to dashboard |
| **REQ-2** | User should be able to register their own account |
| **REQ-3** | Team member shall be able to login and redirect to specific dashboard |

Table 21: Functional requirements for Authorization levels

| Req. No. | Requirement Description |
|---|---|
| **REQ-4** | Users shall be able to assign their own role |

| | |
|---|---|
| **REQ-5** | Users shall have access to the relevant modules associated with their role after logging in |

#### 3.3.3.2 User profile Management

Table 22: Functional requirements for User Profile Management

| Req. No. | Requirement Description |
|---|---|
| **REQ-6** | Users should be able to update their profile |
| **REQ-7** | Users shall be able to delete their account |

#### 3.3.3.3 AI and Machine Learning Integration

Table 23: Functional requirements for AI and Machine Learning Integration

| Req. No. | Requirement Description |
|---|---|
| **REQ-8** | The AI shall enhance and suggest additional relevant details based on the initial bug description provided by the user |
| **REQ-9** | The system shall allow user to accept, reject, or modify AI-generated suggestions before finalizing the bug description |
| **REQ-10** | The Machine Learning model shall analyze the bug description and assign a severity level (e.g., Minor, Major, Critical etc.) |
| **REQ-11** | The system shall display the predicted severity level to the user |
| **REQ-12** | The system shall continuously improve the accuracy of severity predictions by learning from user over time |
| **REQ-13** | The Machine Learning model shall analyze bug details to detect potential relationships (e.g., duplicates, dependencies, or root causes) |
| **REQ-14** | The Machine Learning model shall learn from user feedback on identified relationships to continuously improve its accuracy in finding related bugs. |

#### 3.3.3.4 Project Management and Bug Reporting

Table 24: Functional requirements for Project Management and Bug Reporting

| Req. No. | Requirement Description |
|---|---|
| **REQ-15** | Team member shall be able to manage bug (Create, View, Update and Delete) details |

| | |
|---|---|
| **REQ-16** | Team member shall be able to manage (Create, View, Update and Delete) project details |
| **REQ-17** | Team member shall be able to manage (Create, View, Update and Delete) project team |
| **REQ-18** | The system shall allow a project to be divided into sprints |
| **REQ-19** | The system shall allow user to manage (Create, View, Update and Delete) project sprints |
| **REQ-20** | Team member shall be able to view assigned projects |
| **REQ-21** | Team member shall be able to update the status of the reported bug |
| **REQ-22** | Team Member shall be able to search reported bug |
| **REQ-23** | The system shall allow team member to comment and add attachments to bug discussion |
| **REQ-24** | The system shall allow team members to receive notification upon status of a bug is changed |
| **REQ-25** | The system shall allow team member to generate reports and insights for each project and phases |
| **REQ-26** | The system shall allow team member to move bugs from one sprint to another |

#### 3.3.3.5 Project Summary

Table 25: Functional requirements for Statistic and Reporting

| Req. No. | Requirement Description |
|---|---|
| **REQ-27** | The system shall allow team member to generate reports and insights for each project, team and sprints |

### 3.3.4 Non-Functional requirement

The non-functional requirements for the system focus on key attributes that ensure the system's overall quality and long-term viability. These include usability and maintainability. These non-functional requirements are essential for delivering a robust, user-friendly, and sustainable system that meets both current and future needs.

#### 3.3.4.1 Usability

1. The system should be easy to navigate and adhere to the same standards as other web applications.

#### 3.3.4.2 Maintainability

1. The system should have well-organized file structures, well-documented source code, and be easy to maintain.

## 3.4 System Design

This section outlines the system design of the proposed solution. System design involves defining the architecture, components, modules, interfaces, and data flow to meet the system's requirements. Key design elements such as the Entity-Relationship Diagram (ERD), UML Class Diagram, Activity Diagram, and User Interface (UI) designs will be presented to illustrate the system’s structure and functionality.

### 3.4.1 ERD Diagram

An Entity-Relationship Diagram (ERD) is a visual representation of the data model of a system, illustrating the entities involved and their relationships. It is used to depict how data is structured and connected within a database. In this system, the ERD includes several key entities: User, Bug, Comment, Attachment, Team, TeamMember, Project, ProjectTeam, and Sprint. Figure 11 shows the ERD of the proposed system.

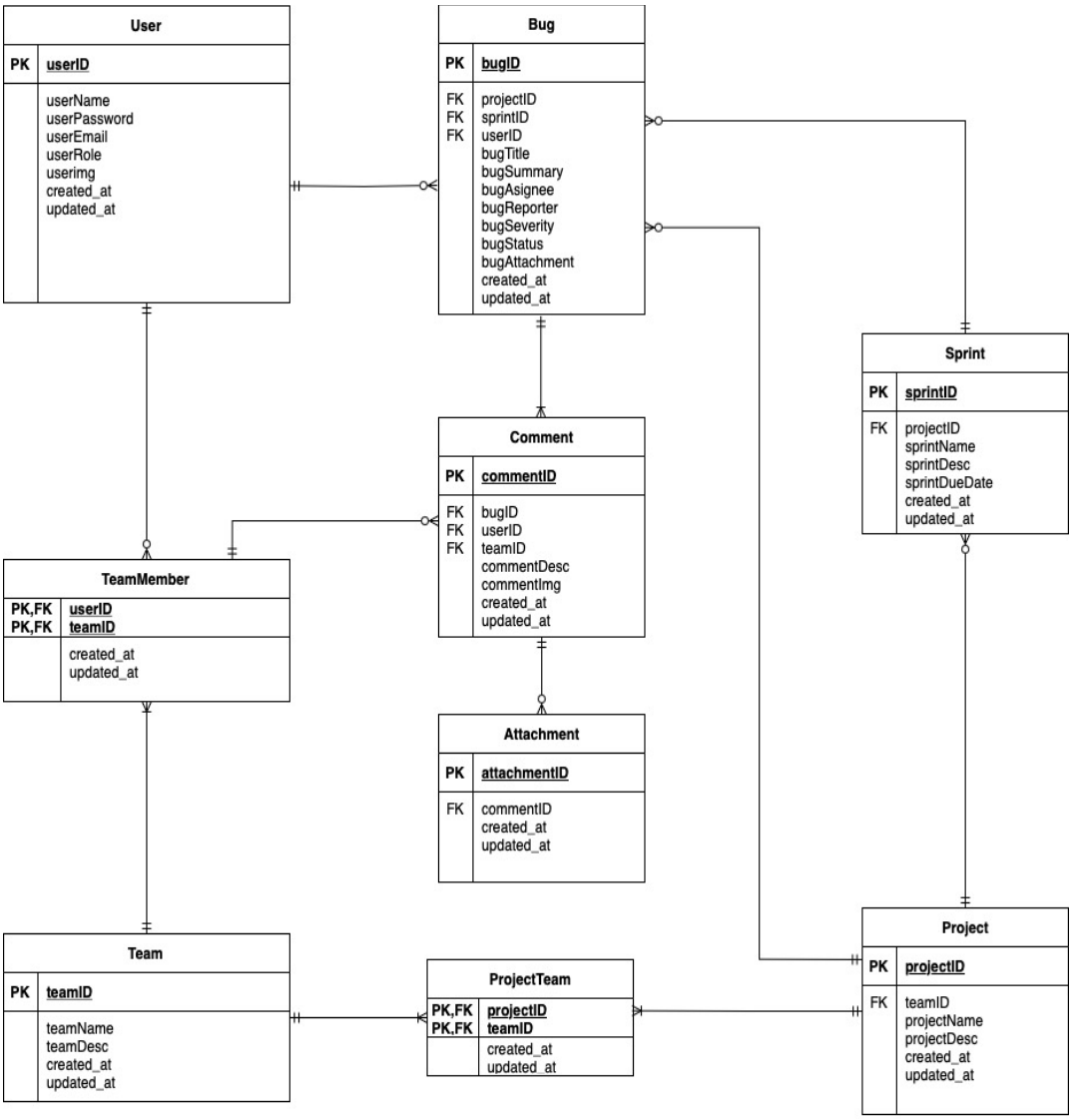



Figure 11: Entity Relationship Diagram

### 3.4.2 UML Class Diagram

A UML class diagram provides a static representation of the structure of a system by illustrating its classes, attributes, methods, and the relationships between them. It defines how different classes interact, highlighting associations, inheritance, and dependencies. The classes include User, Bug, Comment, Attachment, Team, TeamMember, Project, Sprint and ProjectSummary. Figure 12 below shows the UML Class Diagram of the proposed system.

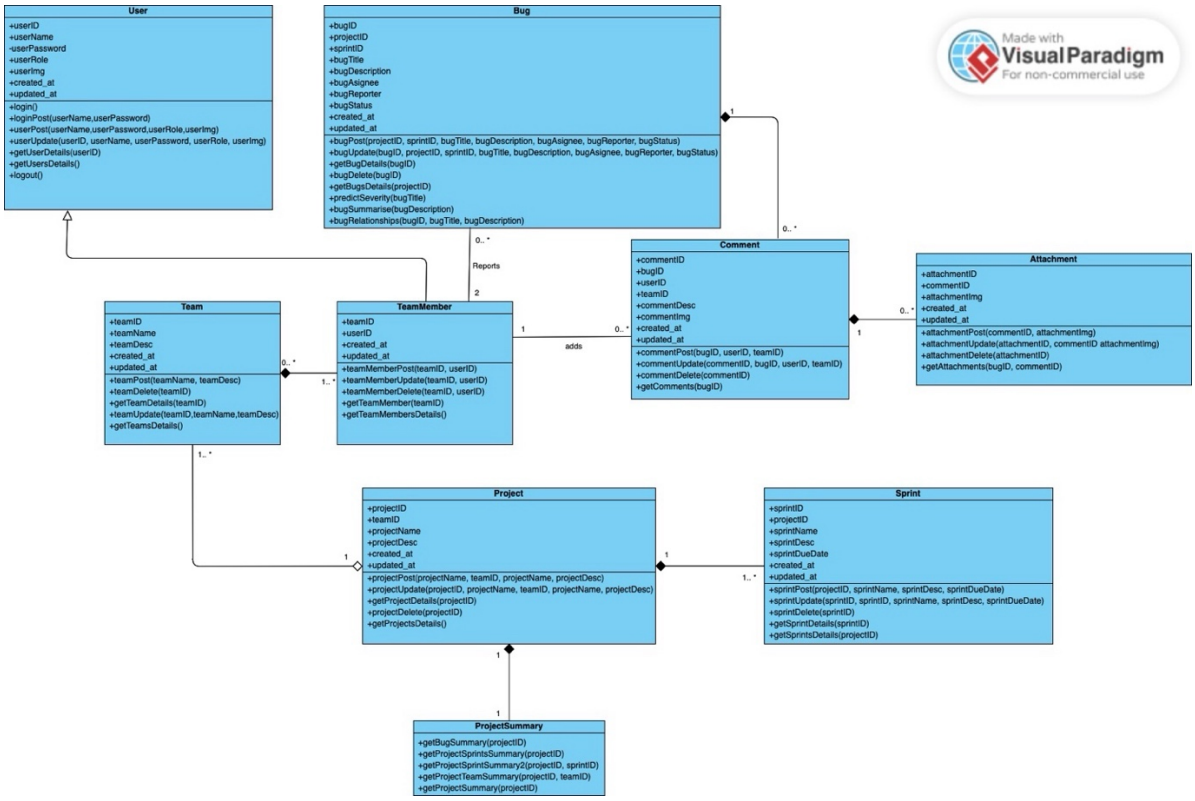



Figure 12: UML Class Diagram

The classes are identified by analyzing of both functional and non-functional requirements gathered during the Inception and Elaboration phases. This included reviewing use cases, user feedback, and system functionality needs. Potential classes were identified by extracting noun phrases from these requirements.

### 3.4.3 Activity Diagram

The activity diagram is a vital tool in modeling the dynamic aspects of the AI-powered Bug Tracking System. It provides a visual representation of workflows and processes, showcasing the sequence of activities, decision points, and interactions within the system. The diagram captures the flow of actions involved in key functionalities, such as reporting a bug, create a project, and assigning it to a team, and resolving it. Figures 13 and Figure 14 present the activity diagrams for the bug reporting and project creation processes, respectively.

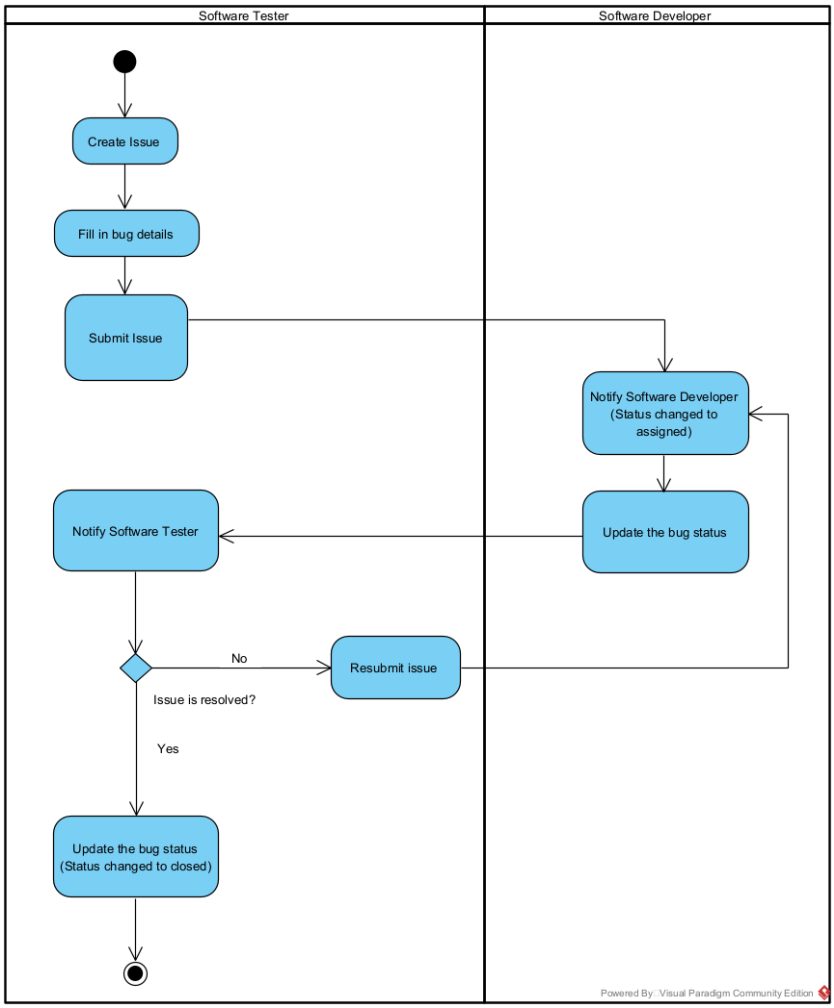



Figure 13: Activity Diagram for Bug Reporting

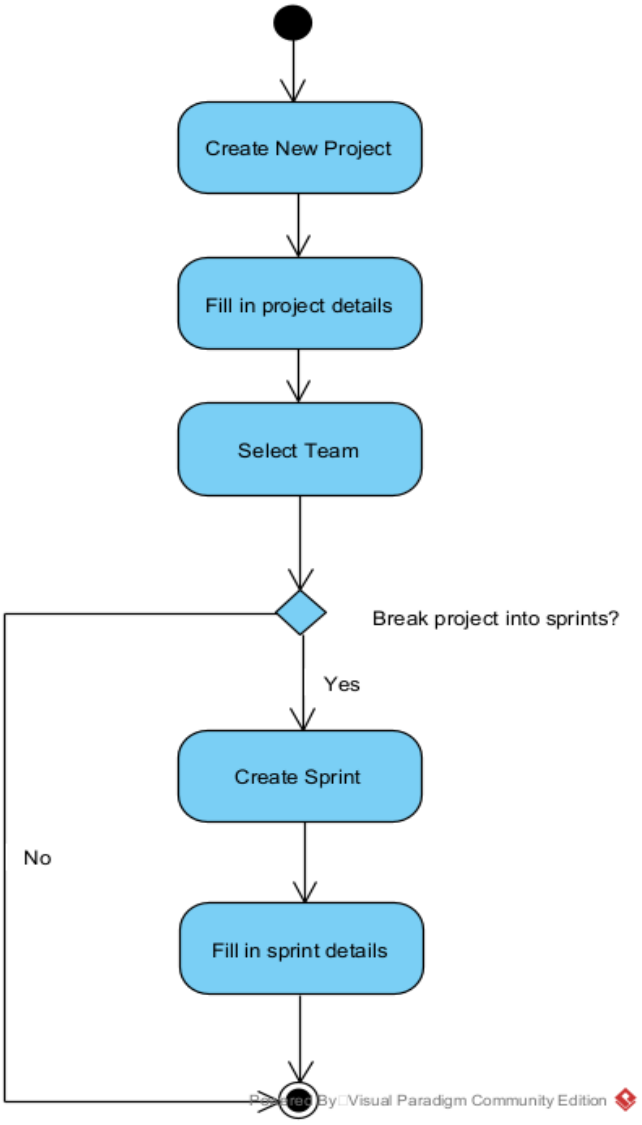



Figure 14: Activity Diagram for Create Project

### 3.4.4 User Interface Design

The user interface (UI) design is a critical aspect of the AI-powered Bug Tracking System, as it directly impacts user experience and system usability. To ensure the interface aligns with user expectations and operational workflows, the design process was conducted iteratively with the user during elaboration phase. Regular consultations and feedback sessions were held to gather insights into their specific requirements and preferences. The system user interface will cover all functions including login page (Figure 15), dashboard page (Figure 16), View bugs page (Figure 17), View bug page (Figure 18), create bug page (Figure 19), create new project page (Figure 20), view projects page (Figure 21), edit project page (Figure 22), project summary page (Figure 23), project sprints page (Figure 24), create new project sprints page (Figure 25), edit project sprint page (Figure 26), edit profile page (Figure 27) , view users page (Figure 28), add users page (Figure 29), edit users details page (Figure 30), view teams page (Figure 31), create new team page (Figure 32), and edit team details page (Figure 33).

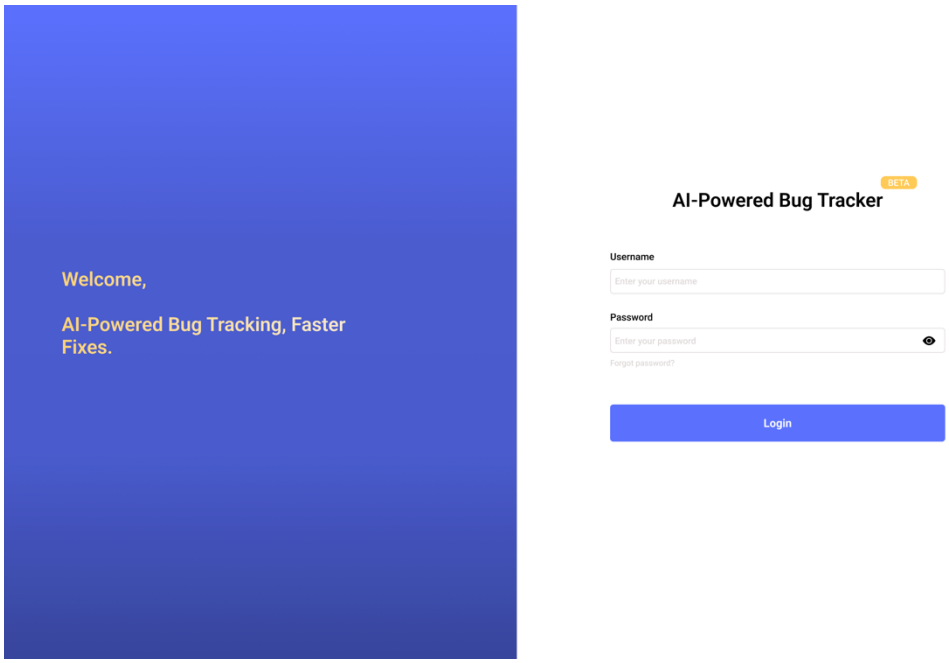



Figure 15: Login Page

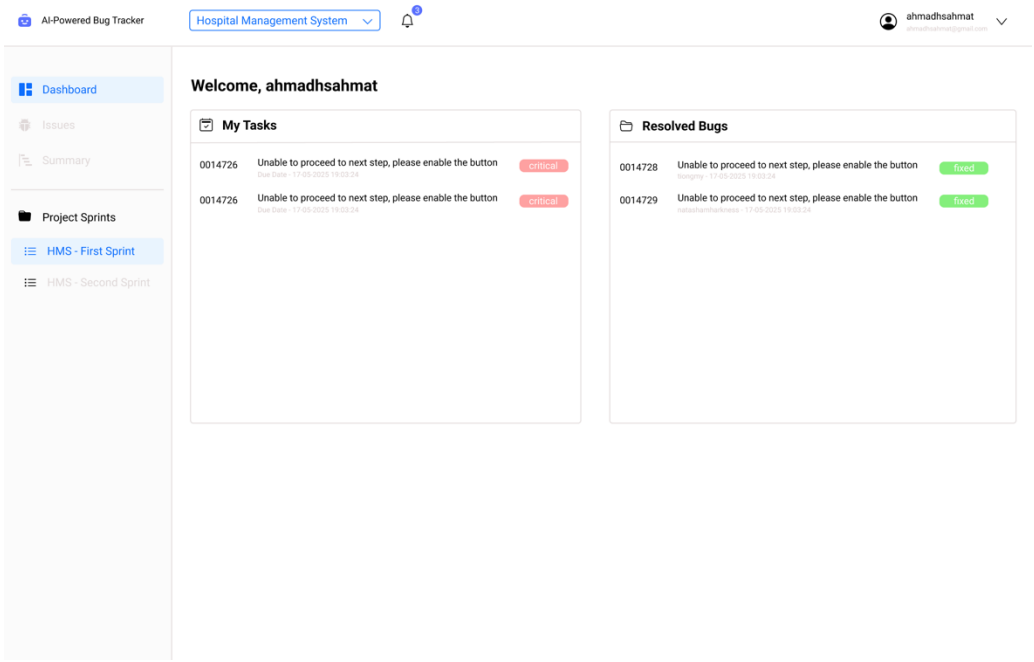



Figure 16: Dashboard Page

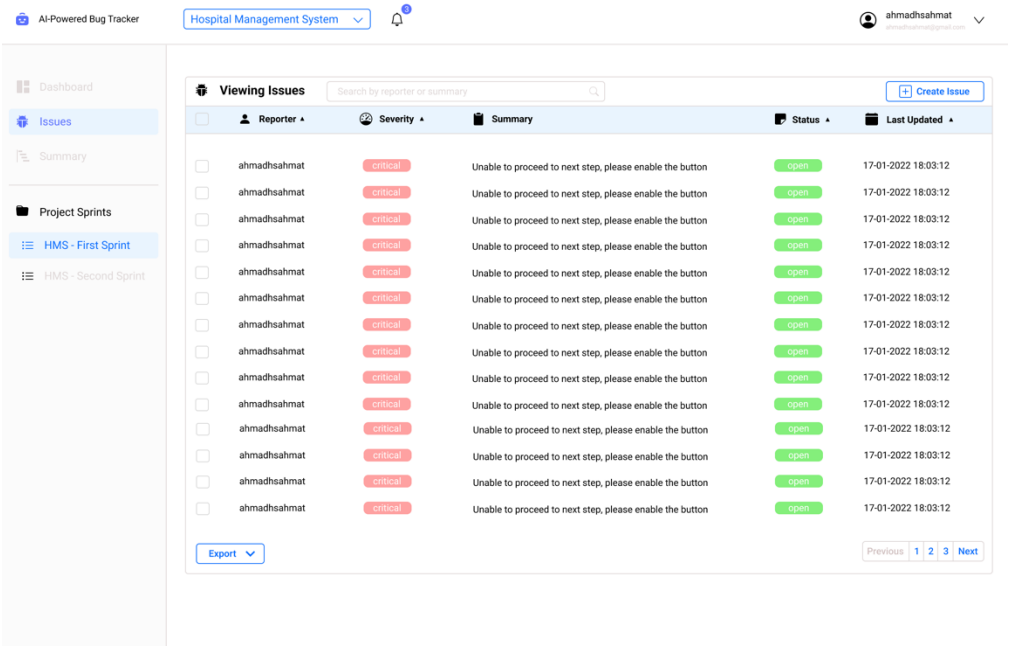



Figure 17: View Bugs Page

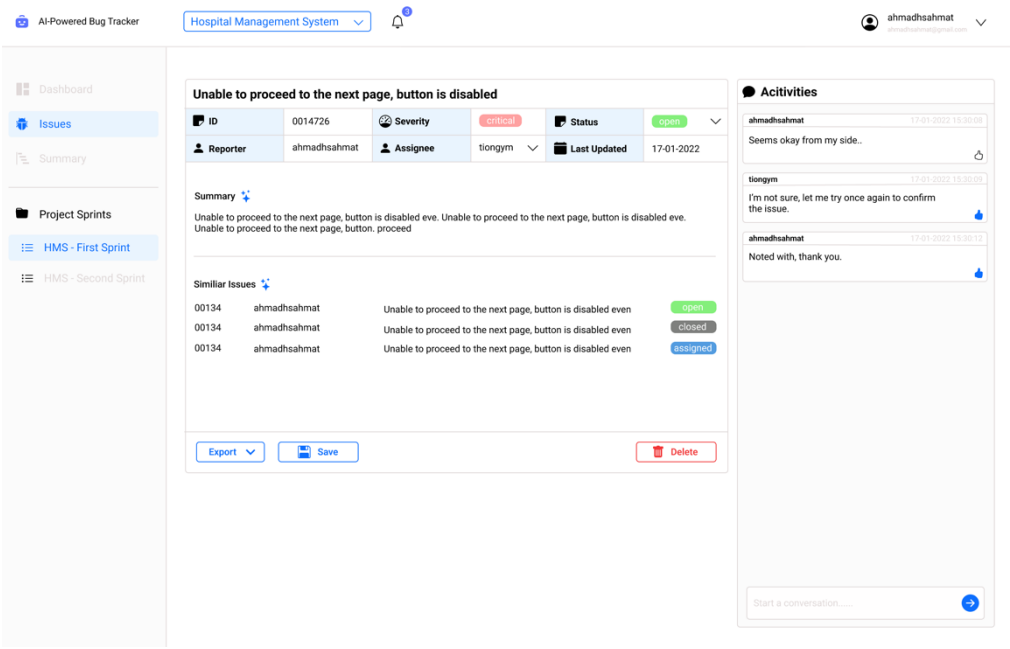



Figure 18: View Bug Page

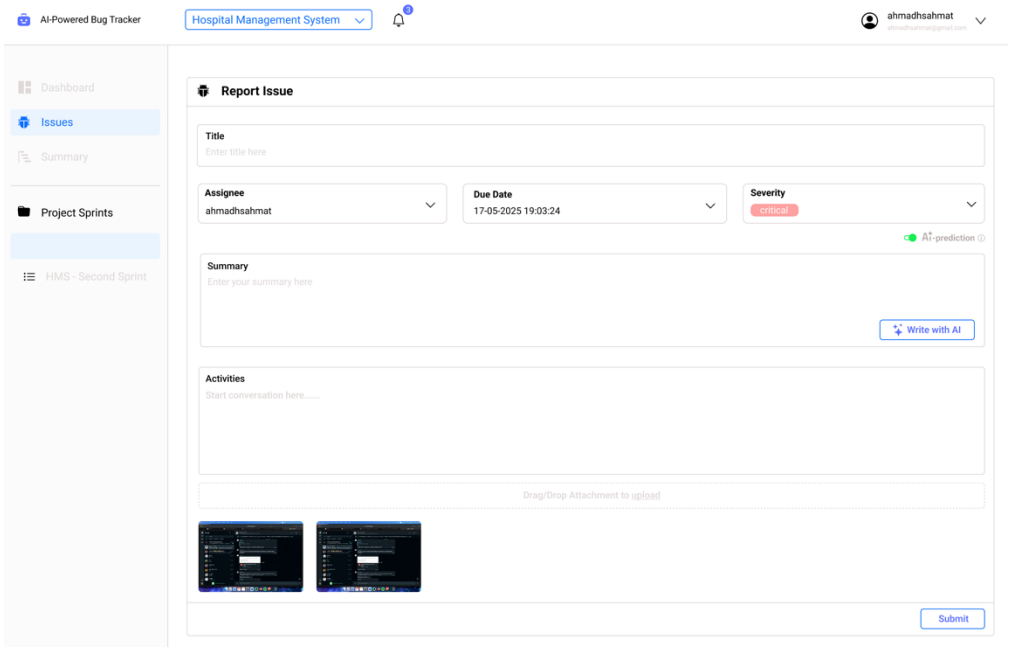



Figure 19: Create Bug Page

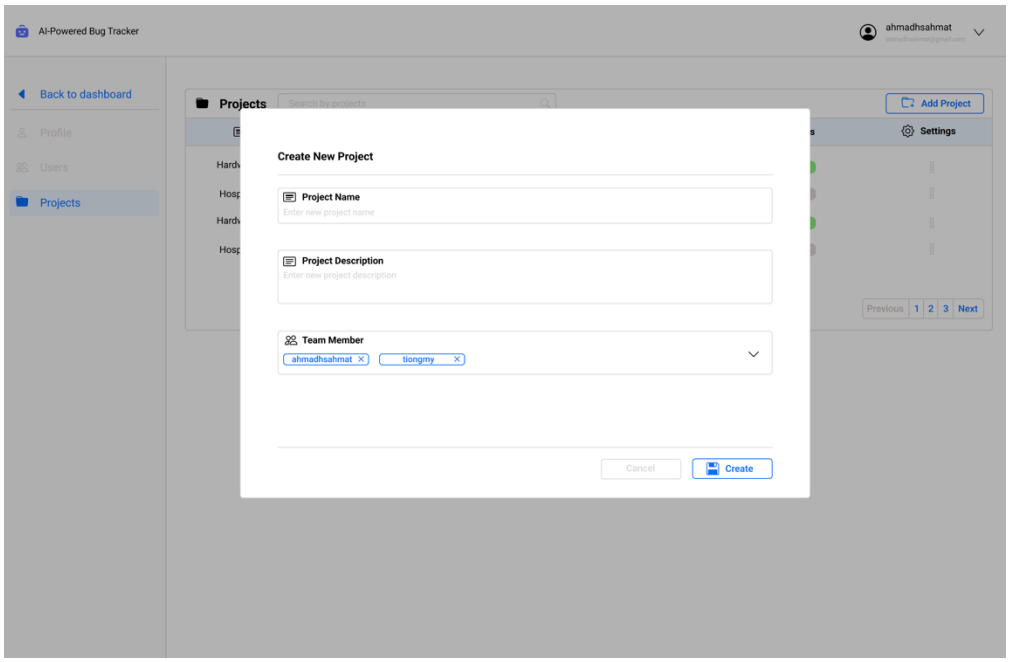



Figure 20: Create New Project Page

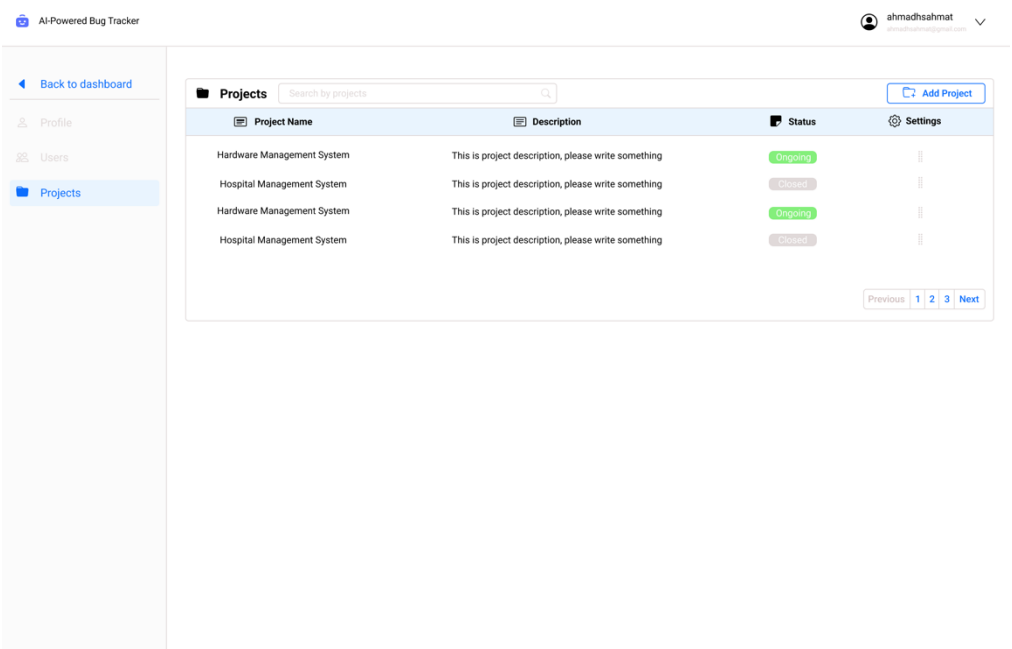



Figure 21: View Projects Page

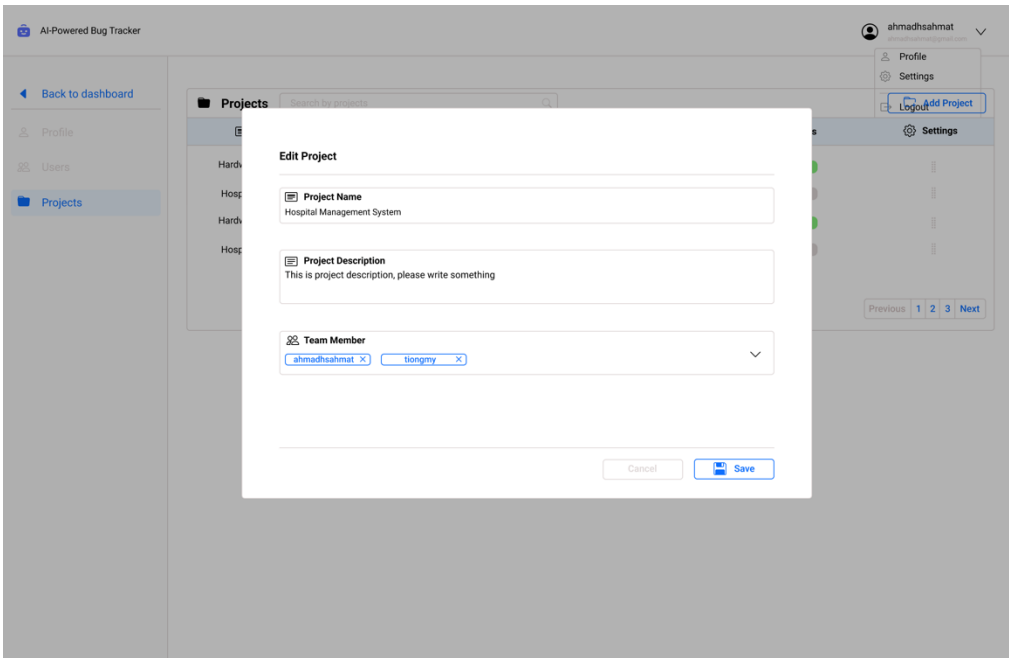



Figure 22: Edit Project Page

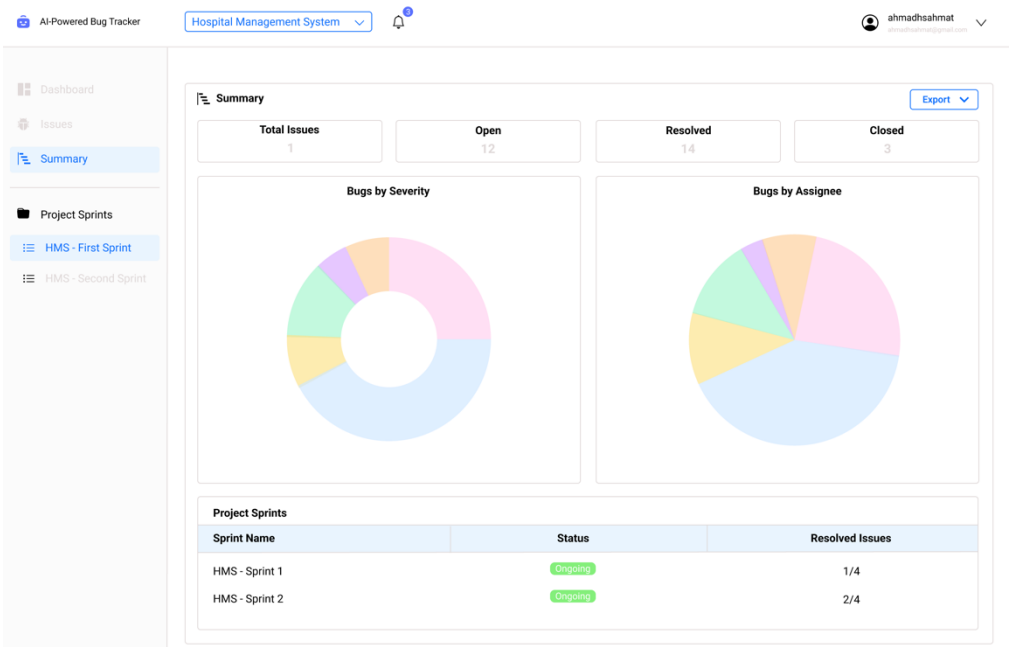



Figure 23: Project Summary Page

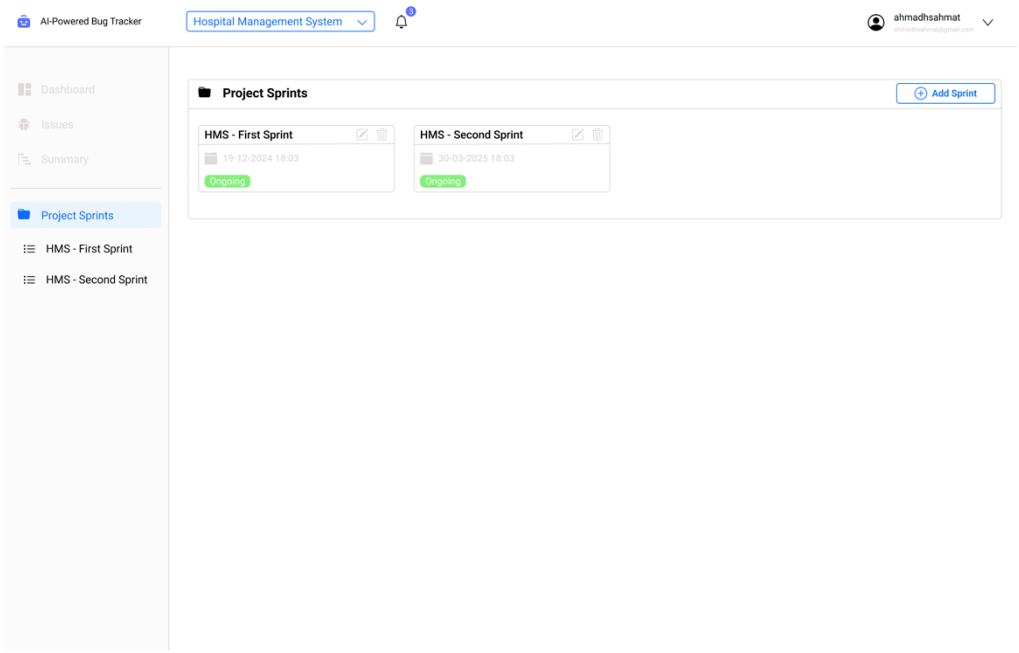



Figure 24: Project Sprints Page

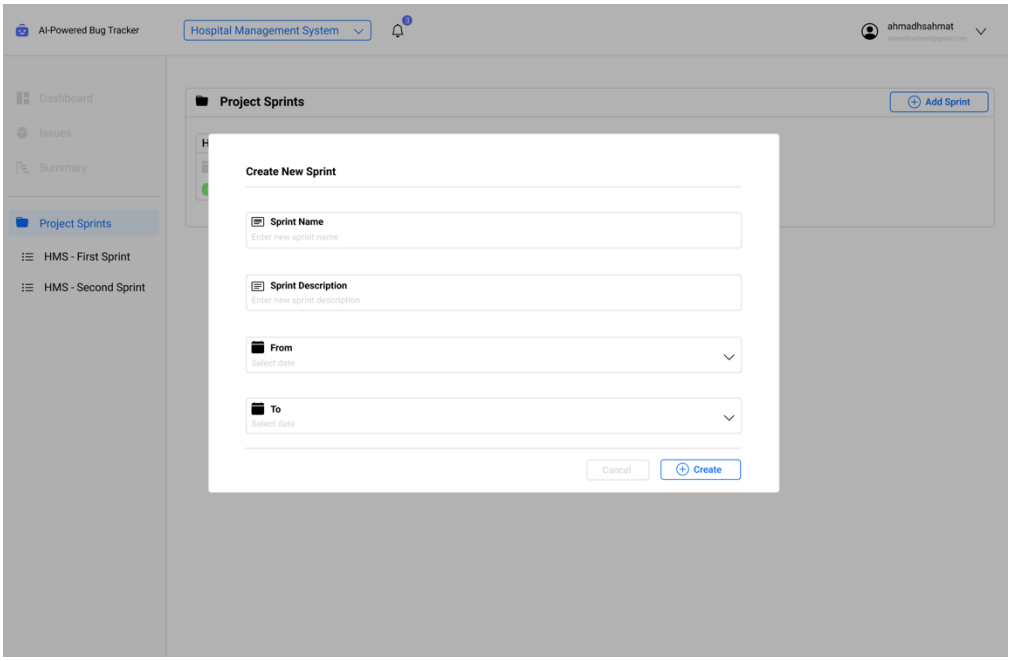



Figure 25: Create New Project Sprint

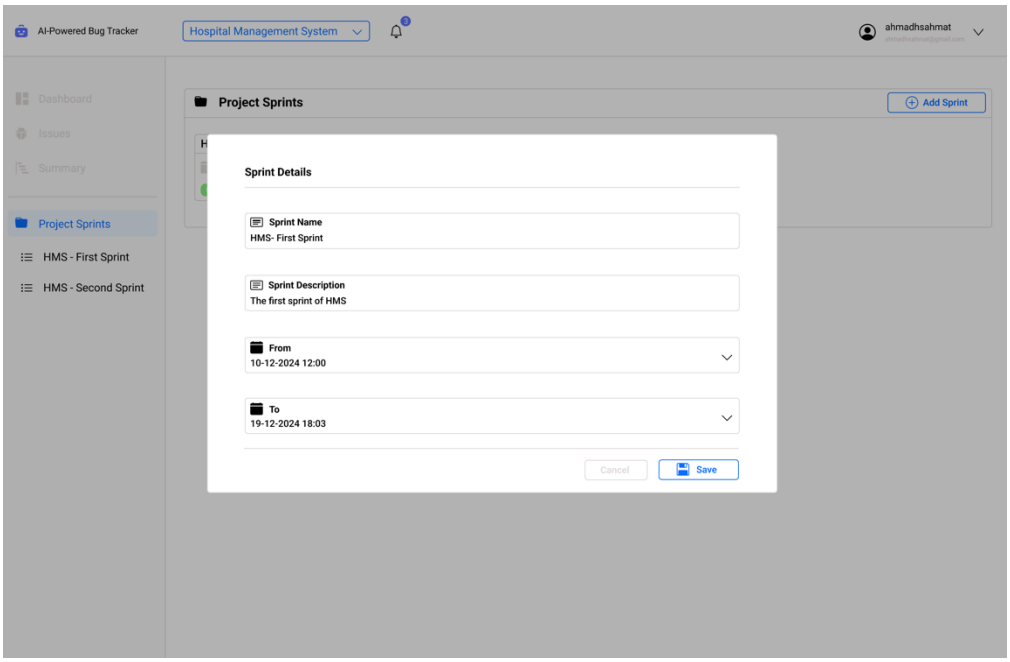



Figure 26: Edit Project Sprint

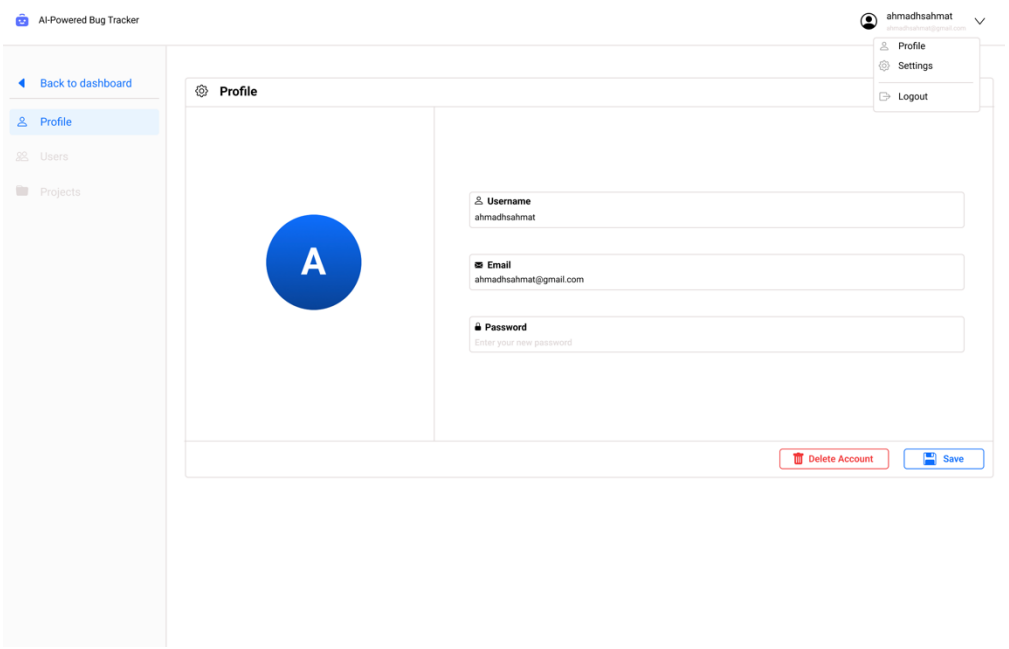



Figure 27: Edit Profile Page

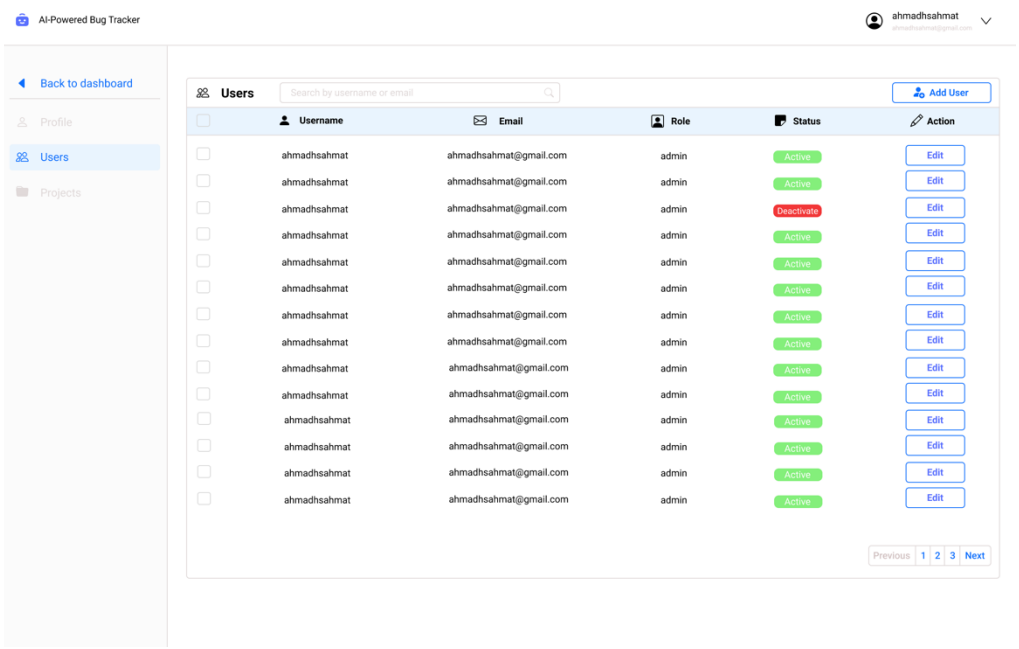



Figure 28: View Users Page

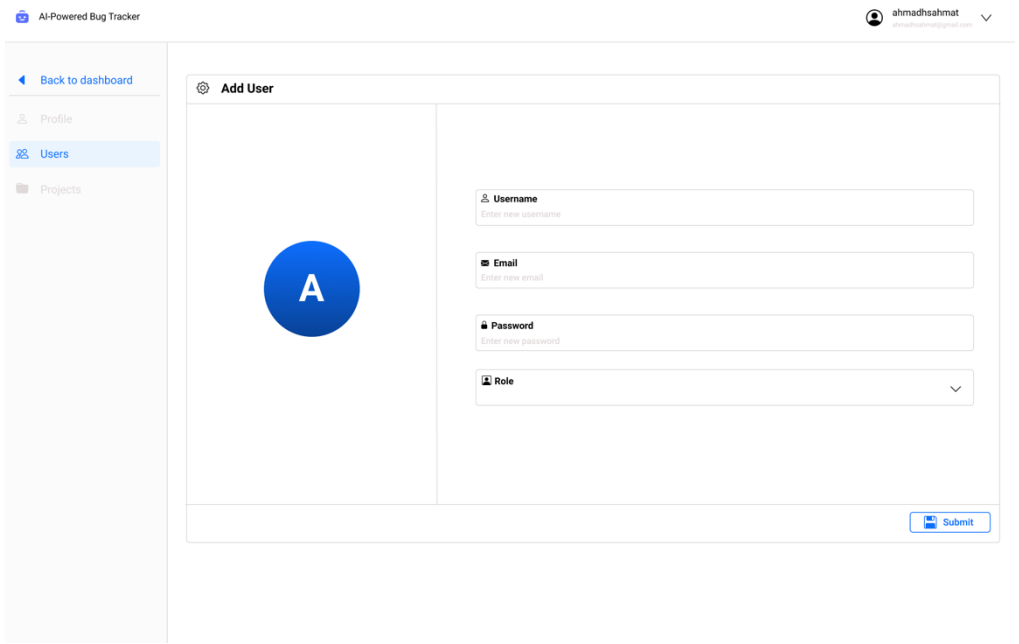



Figure 29: Add User Page

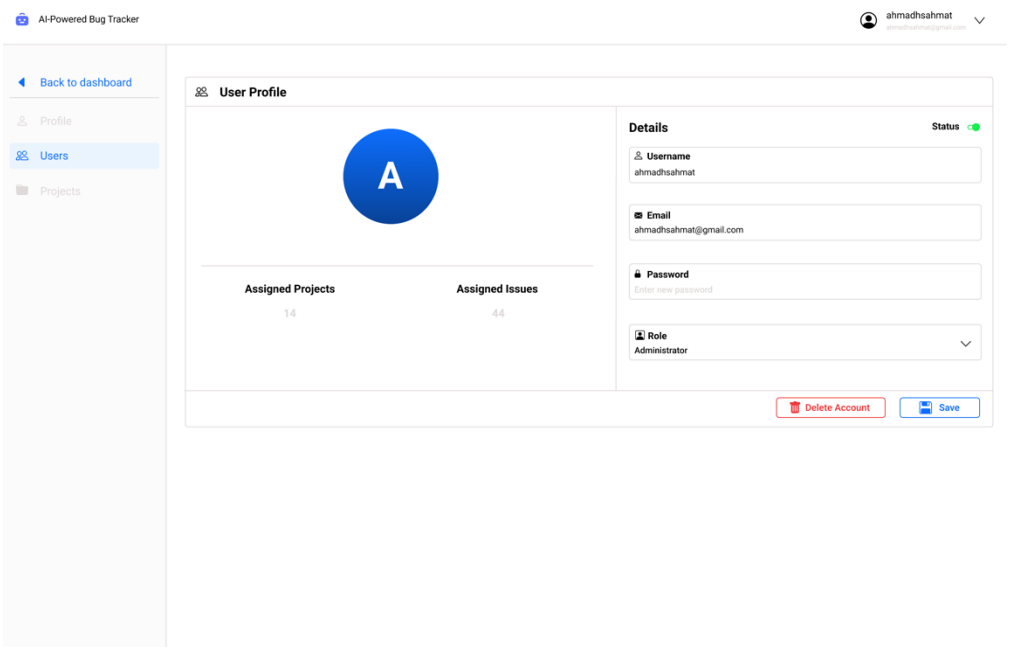



Figure 30: Edit User Details Page

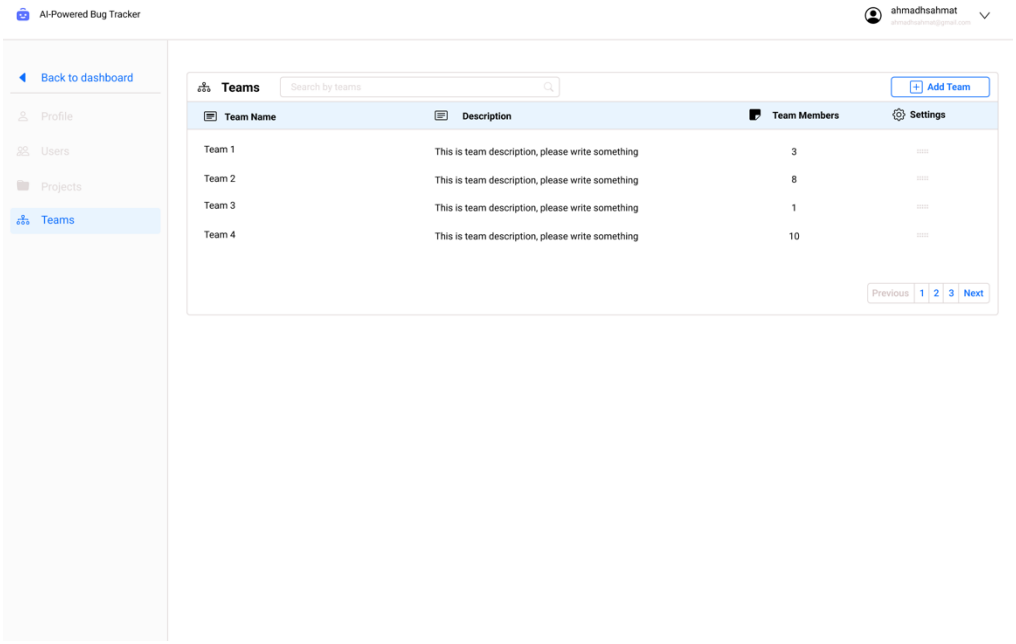



Figure 31: View Teams Page

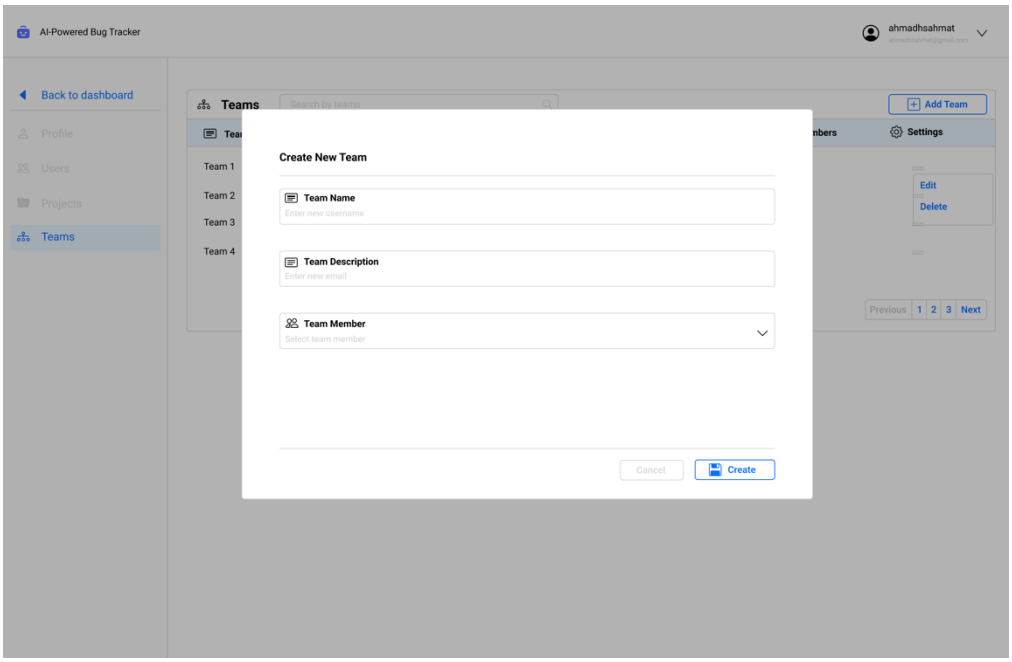



Figure 32: Create New Team Page

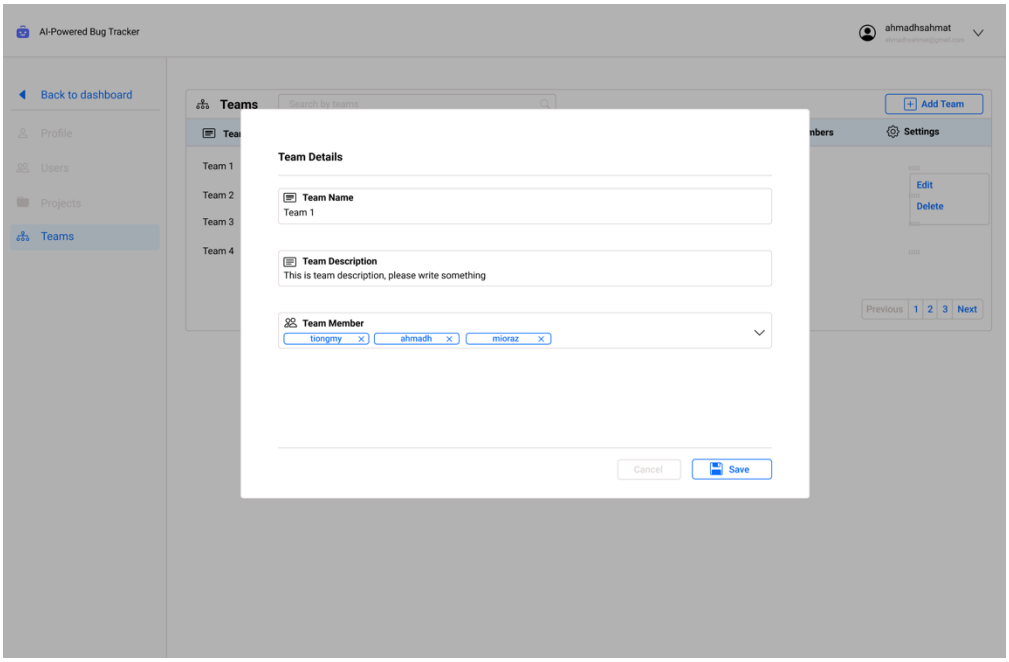



Figure 33: Edit Team Details Page

The user interface design underwent an iterative process, beginning with the presentation of the initial design to user. Feedback from user during this stage, which primarily emphasized the

need for an easy-to-use interface, a modern aesthetic, and inclusivity was carefully analyzed to address usability concerns and align the interface with user expectations. By refining the design through user comments, the final interface achieves greater functionality, intuitiveness, and user satisfaction.

## 3.5 Summary

This chapter covers the process of requirements gathering, conceptual design, and user interface of the system development during inception and elaboration phases. All necessary requirements have been collected, and the conceptual design, along with the user interface, has been developed to ensure clarity and direction during the construction phase. However, the requirements and system design may be subject to change, as sudden change requests might occur during construction phase. These steps are to provide a solid foundation, ensuring that the system will be built in alignment with the defined goals and user needs, while remaining adaptable to any unforeseen adjustments.

# CHAPTER 4: IMPLEMENTATION

## 4.1 Introduction

This chapter outlines the implementation of the AI-powered bug tracking tool. It begins by introducing the software tools and technologies used to support the implementation of the proposed system based on the design and functionality. Following this, the chapter explains the roles and permissions assigned to users across various modules of the system. It also explains each component in detail, highlighting its functionality and the way users will interact with the interface. Together, these elements provide a comprehensive understanding of the system's architecture and how it will operate in a real-world environment.

## 4.2 Installation and Configuration of the Software/Tools Used

The implementation begins with setting up the development environment to ensure all necessary tools and resources are available for building the AI-powered bug tracking tool. The table below outlines the IDEs and Software Tools required to effectively develop the proposed system and these tools will help to streamline the software development process. Table 26 shows the software tools and IDE used in this project.

Table 26: Software Tools and IDE

| | | |
|---|---|---|
| **Software Tools** | Laravel Herd, MySQL Workbench | |
| **Integrated Development Environment (IDE)** | Web Development: | Visual Studio Code |
| | Machine Learning: | Python, jupyter |
| **Front-end Frameworks** | ReactJs | |

| | |
|---|---|
| **Back-end Frameworks** | Laravel, Flask |
| **Cloud Services** | Azure Services |

### 4.2.1 Laravel Herd

Laravel Herd provides a user-friendly interface that allows users to create projects without the need for terminal commands. Its primary purpose is to host websites locally during development, simplifying the process of setting up and managing local environments. Figure 34 shows the interface of the Laravel Herd software tool.

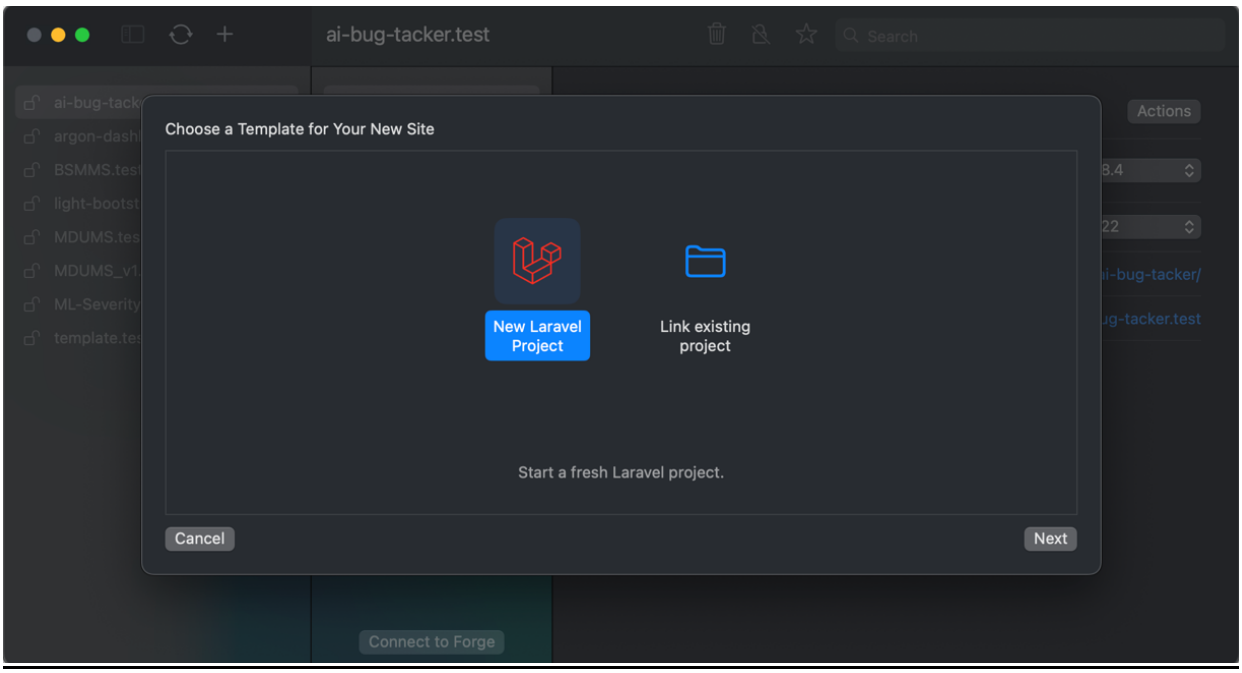



Figure 34: Laravel Herd Interface

### 4.2.2 MySQL Workbench

MySQL Workbench is a powerful tool for managing the system's database. It offers a user-friendly graphical interface (GUI) that simplifies database administration tasks. Additionally, MySQL Workbench provides fast SQL query execution, making it an essential tool for developers to efficiently interact with and manage the database. Figure 35 shows the interface of the MySQL Workbench.

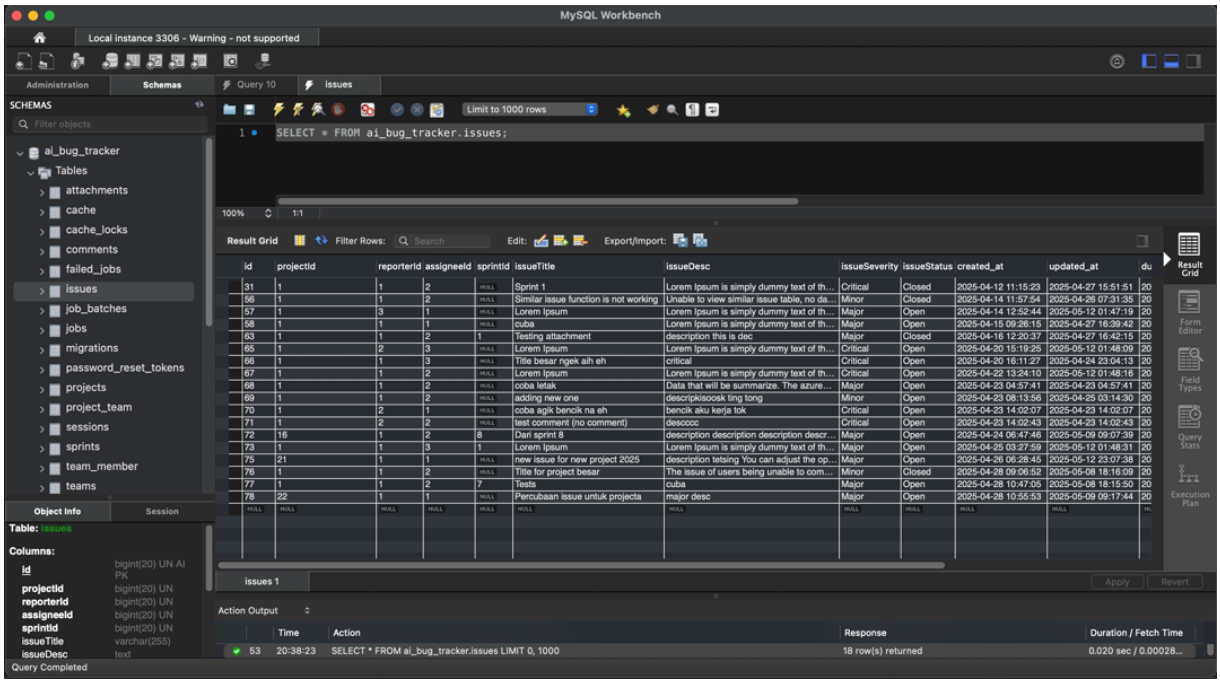



Figure 35: MySQL Workbench

### 4.2.3 Visual Studio Code

Visual studio code is a popular IDE for software development. In this project there are 2 programming languages that will be used to develop the proposed system including PHP and TypeScript. Figure 36 shows the interface of the Visual Studio code IDE.

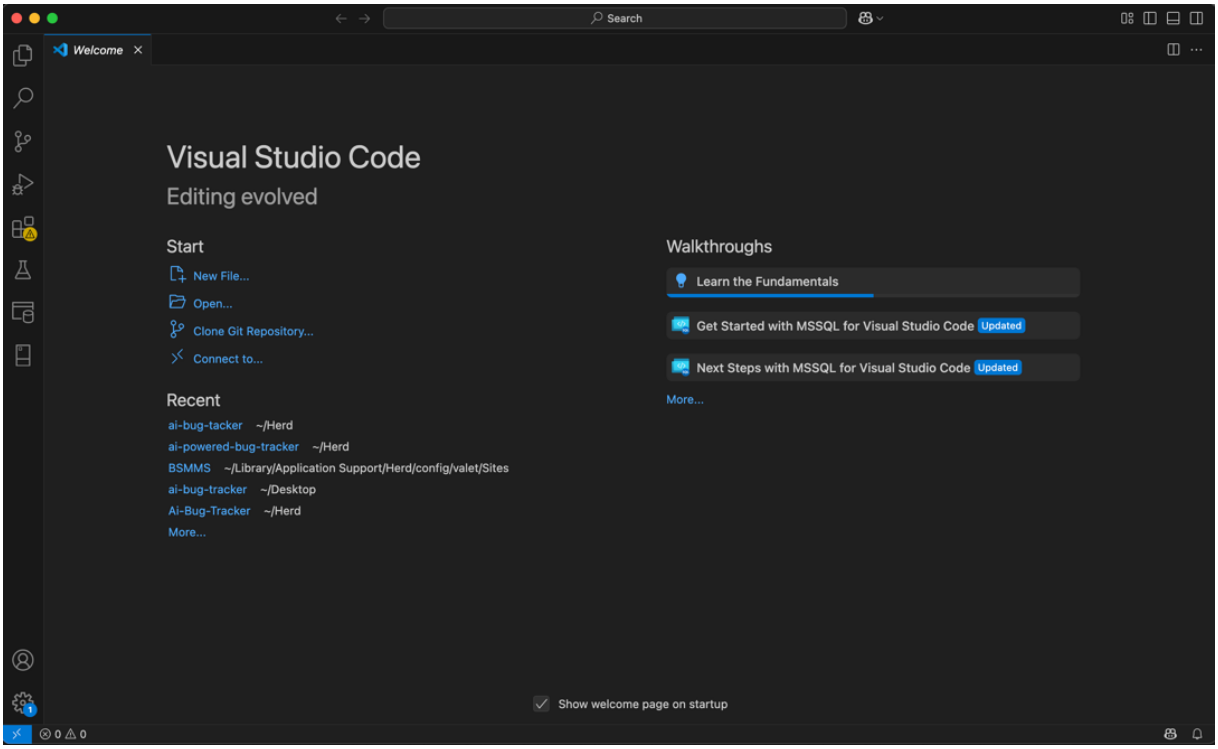



Figure 36: Visual Studio Code interface

### 4.2.4 Jupyter Notebook

Jupyter notebook is a popular IDE for machine learning development and widely used to train, testing and deploy machine learning model. In this project there are 2 machine learning models that will be developed using this tool including bug severity prediction and similar bug detection. Figure 37 shows the interface of the Jupyter Notebook.

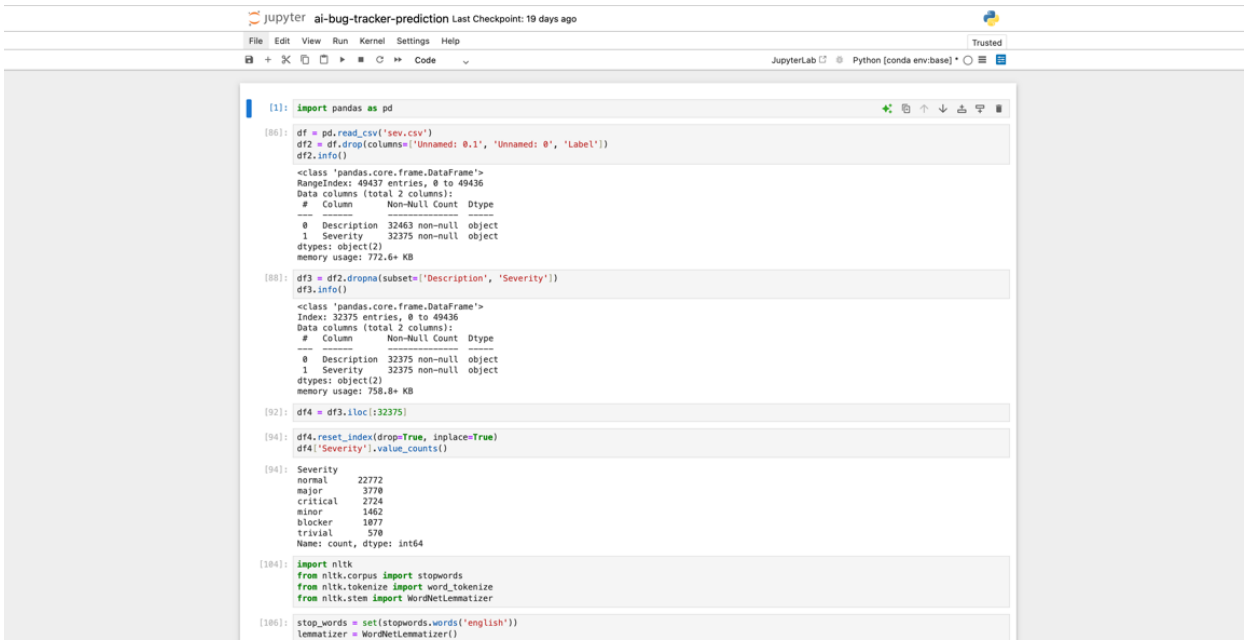



Figure 37: Jupyter notebook interface

## 4.3 Implementation of core features

This section provides an overview of the roles and permissions within the system, along with a detailed explanation of each component and its corresponding system interface**.** Roles and permissions define what actions users can perform, such as managing issues, projects, teams, or other users, ensuring that access and responsibilities are properly controlled. Additionally, this section explores the system interface in depth, describing the design and functionality of each component, and how users interact with them to carry out various tasks within the system.

### 4.3.1 Roles and permissions

The system will have three types of users including software tester, developers and administrator. Software tester and developers mostly share the same level of permissions and access while administrator has a high-level access. Below shows the level of permissions and access for each user.

The software tester and developers able to:

- Login
- View Dashboard
- Manage profile
- Manage projects
- Manage teams
- Manage issues
- View Project and sprint Summary

### 4.3.2 System setup

To run the system, user need to navigate to the main path of the system repository, which is “ai-bug-tracker” file. After successfully navigate to the main file, open terminal and type “npm run dev”. Figure 38 below shows the command prompt to run the system.

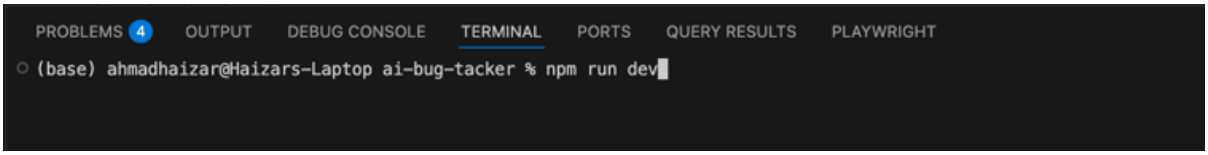



Figure 38: Command to run the system

After running the “npm run dev” command, a web link will be provided at the end of the command line. The generated link will navigate user to the main page of the system, which is the login page. Figure 39 show the generated web link in the terminal.

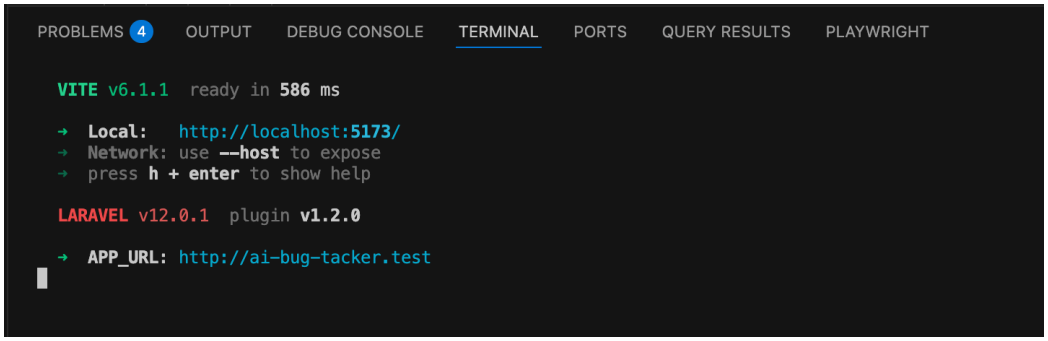



Figure 39: Generated link to access the system

### 4.3.3 System interface

This section provides a comprehensive overview of the system interface, detailing its layout and the functionality of key components. It showcases how users interact with the system through various interface sections, such as Authentication, Forgot Password, Dashboard, Issues,

Projects, Teams, AI integration, and Summary. Each section will be described in detail, highlighting its features, and how users can navigate and perform tasks within the system. This section will offer a thorough explanation of the user interface of each component.

#### 4.3.3.1 Authentication

On the Login Page, users are required to enter their email address and password to access the system. If incorrect credentials are provided, an error message will appear, prompting the user to re-enter the correct information. Upon successful login, users are redirected to their respective dashboard, where they can begin interacting with the system. Additionally, users have the option to enable the “Remember Me” feature, which allows their login credentials to be automatically filled in the future, simplifying the login process when returning to the system. Figure 40 shows the interface of the login page.

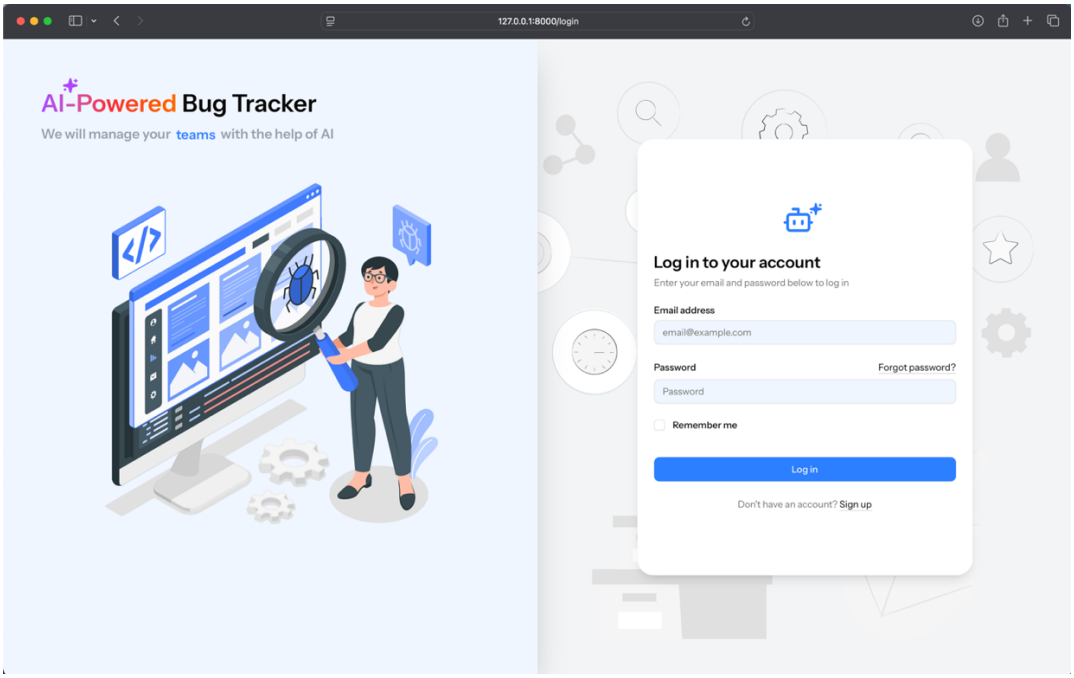



Figure 40: Login Page

#### 4.3.3.2 Forgot Password

The Forgot Password feature allows users to reset their password if they've forgotten it. To begin the process, users simply enter their registered email address. A password reset link will then be sent to their email. By clicking the link, users are securely redirected to a page where they can create a new password, ensuring they can regain access to their account safely. This feature helps maintain security while offering a straightforward way for users to recover their login credentials. Figure 41 shows the user interface for forgot password page.

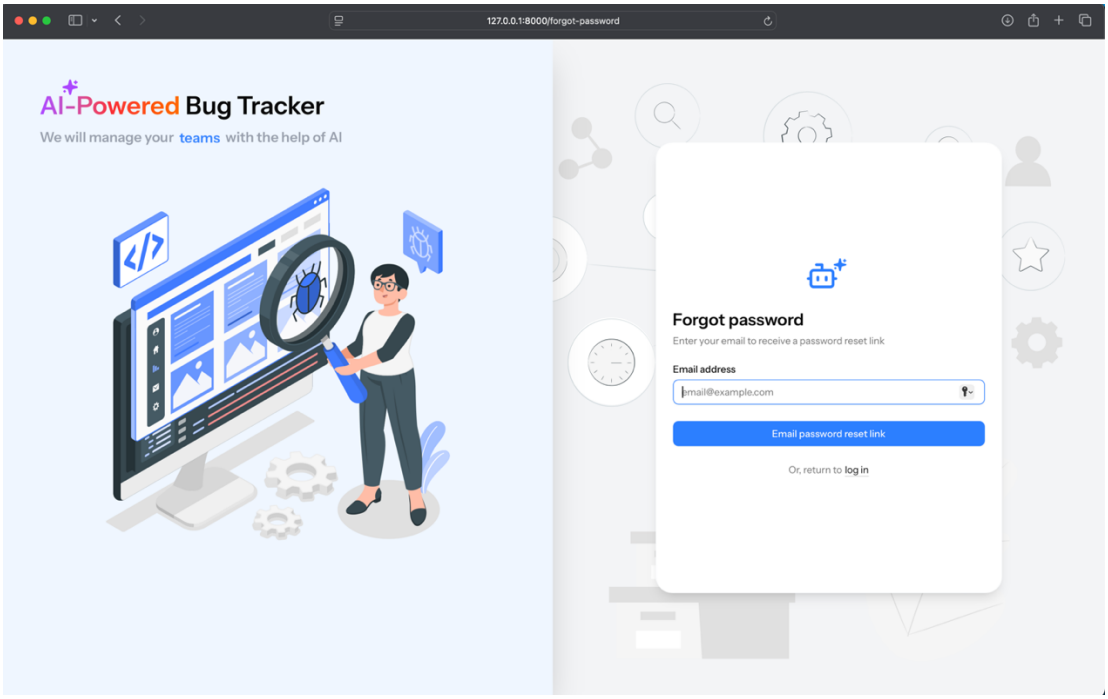



Figure 41: Forgot Password interface

#### 4.3.3.3 Dashboard

The dashboard provides users with a visual overview of their performance and ongoing project activities. Figure 42 below shows the user interface for dashboard. It displays statistics such as

- Total of today's issues
- Ongoing project
- Closed project
- Ongoing issues
- Closed issues
- Latest Activities
- Brief Projects Summary

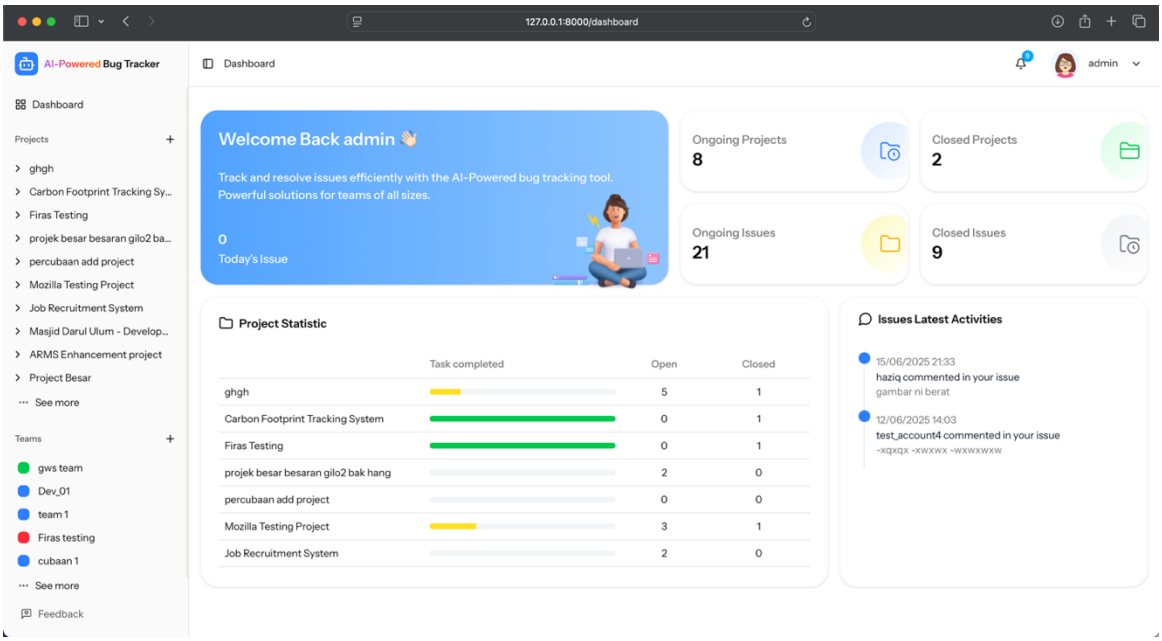



Figure 42: Dashboard interface

#### 4.3.3.4 User profile

User able to update their profile details by navigating to user settings. In this module, user able to manage their profile information and password. Figure 43 shows the user interface of the User profile.

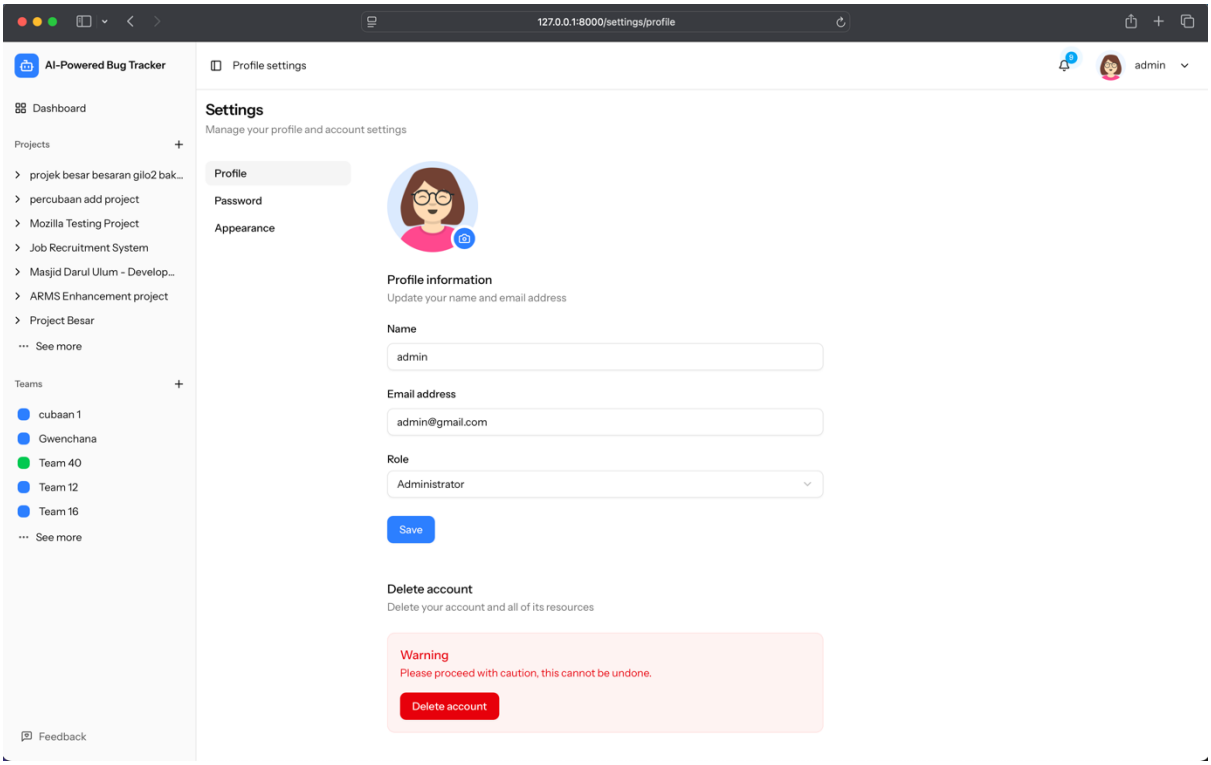



Figure 43: User profile interface

#### 4.3.3.5 Create Issue

Users can create a new issue by clicking the "Create Issue" button, which opens a sheet of form for entering all necessary issue details. Figure 44 below shows the create issue user interface.

1. Title
2. Assignee
3. Due Date
4. Severity
5. Description
6. Root of cause
7. Activities/comments - optional
8. Attachment – optional

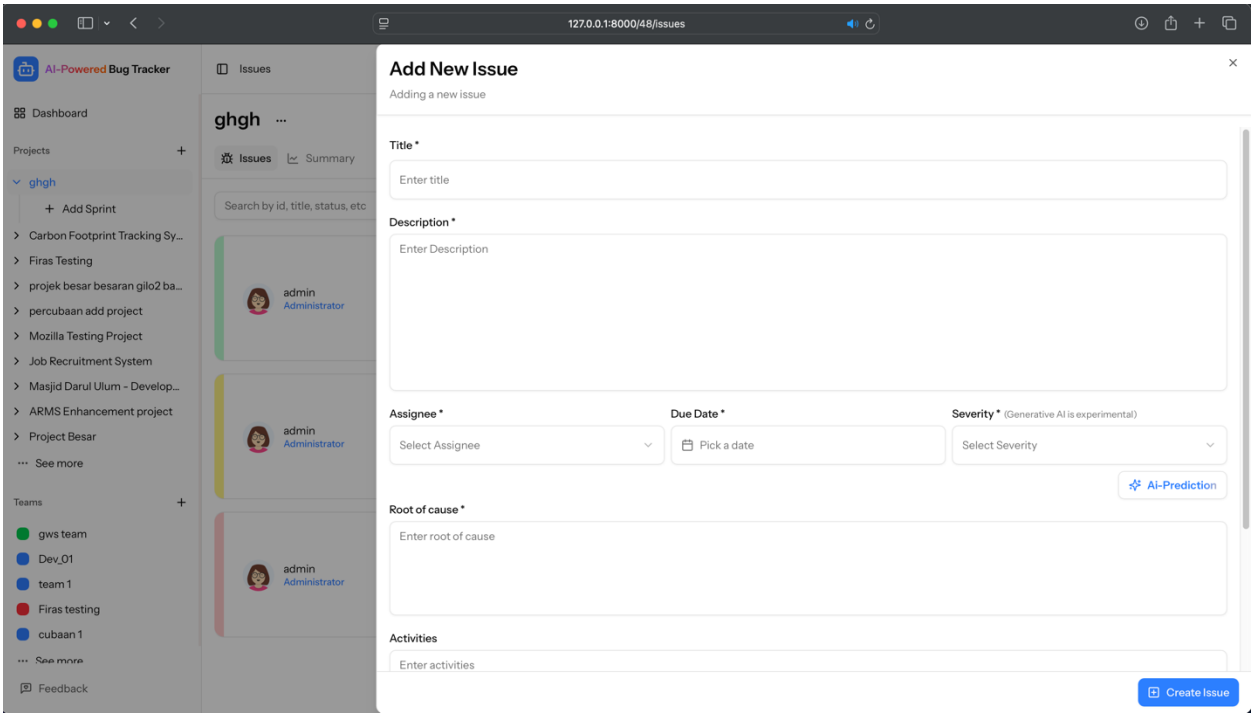



Figure 44: Create issue interface

#### 4.3.3.6 View all issues

In view all issue page, user able to view all issues that has been reported within project. The view all issues will only show the general details of an issue such as the reporter, severity, title, last updated, description, status and total comments. Figure 45 shows the view all issues user interface.

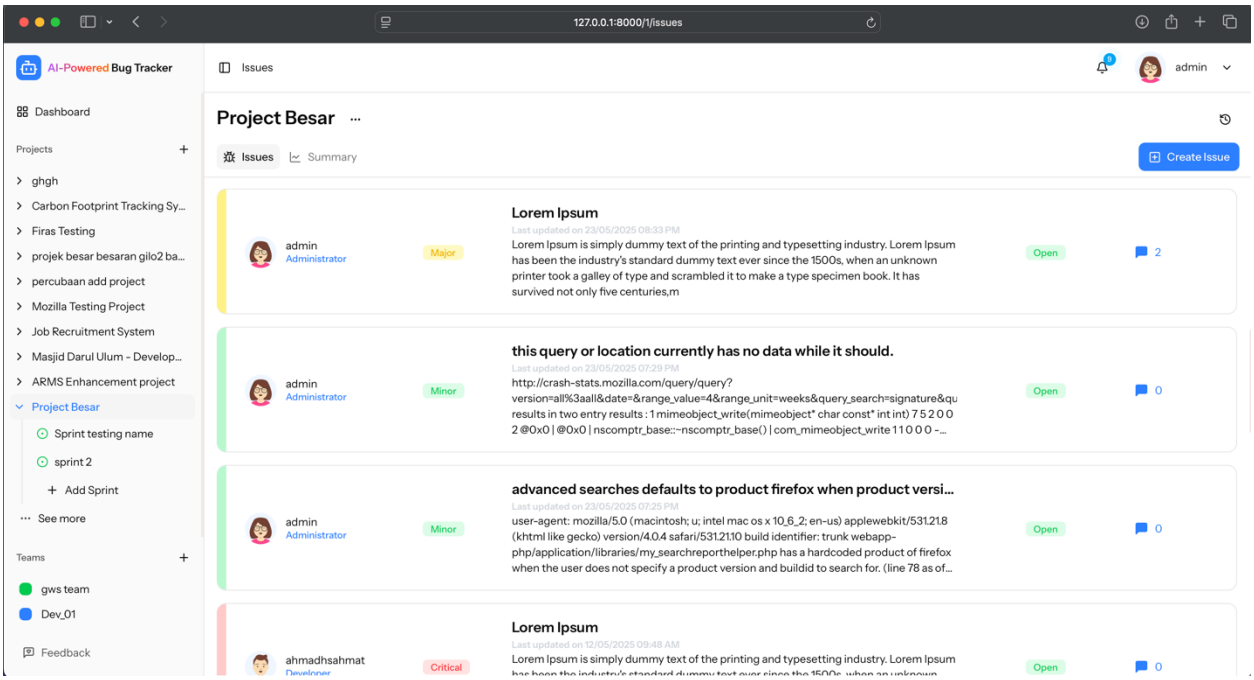



Figure 45: View all issues interface

### 4.3.3.7 View, Update and Delete Issue

Unlike the View All Issues component, which displays a summarized list of issues, the View Issue component provides a detailed view of a single issue. This component shows comprehensive information including the issue title, assignee, last updated date, current status, due date, severity level, detailed description, identified root cause, related activities or comments and its attachments associated. Figure 46 shows the manage issue user interface.

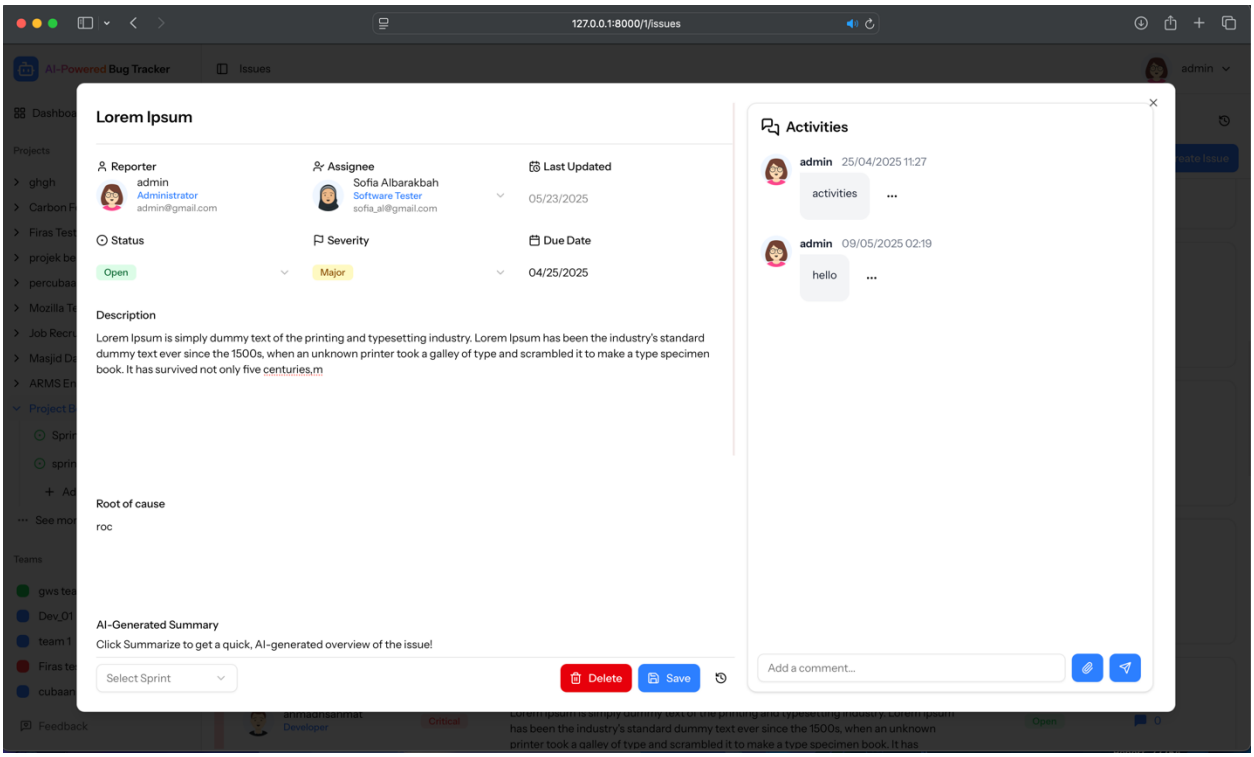



Figure 46: Manage issue interface

#### 4.3.3.8 Create project

Users can create a new project by clicking the "+" button located in the Project section on the sidebar. This opens a form where users are required to fill in and select essential project details, such as title, description and teams. Figure 47 below shows the add new project user interface.

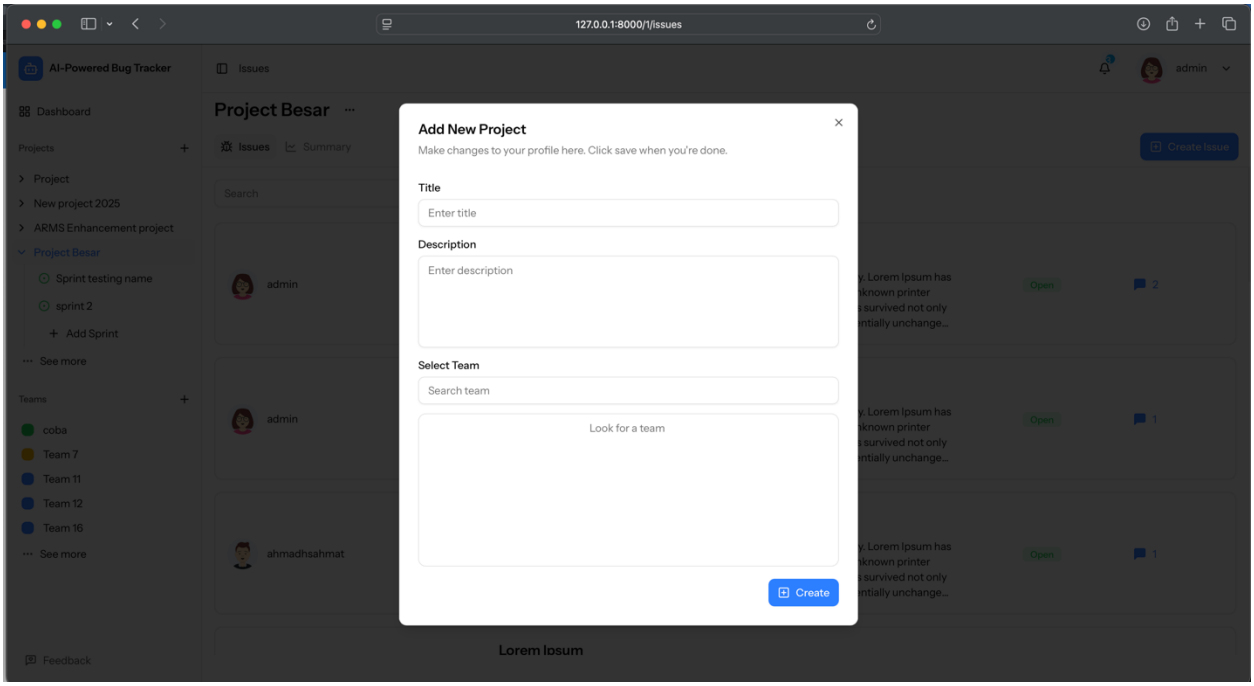



Figure 47: Add new project interface

#### 4.3.3.9 View all projects

In the View All Projects component, users can view a list of all projects they have been assigned to. Each project entry displays key summary information to give users a quick overview of its current state. This includes the project title, description, author, status, total number of issues, the count of open and closed issues, assigned teams, and overall project progress. Figure 48 below shows the view all projects user interface.

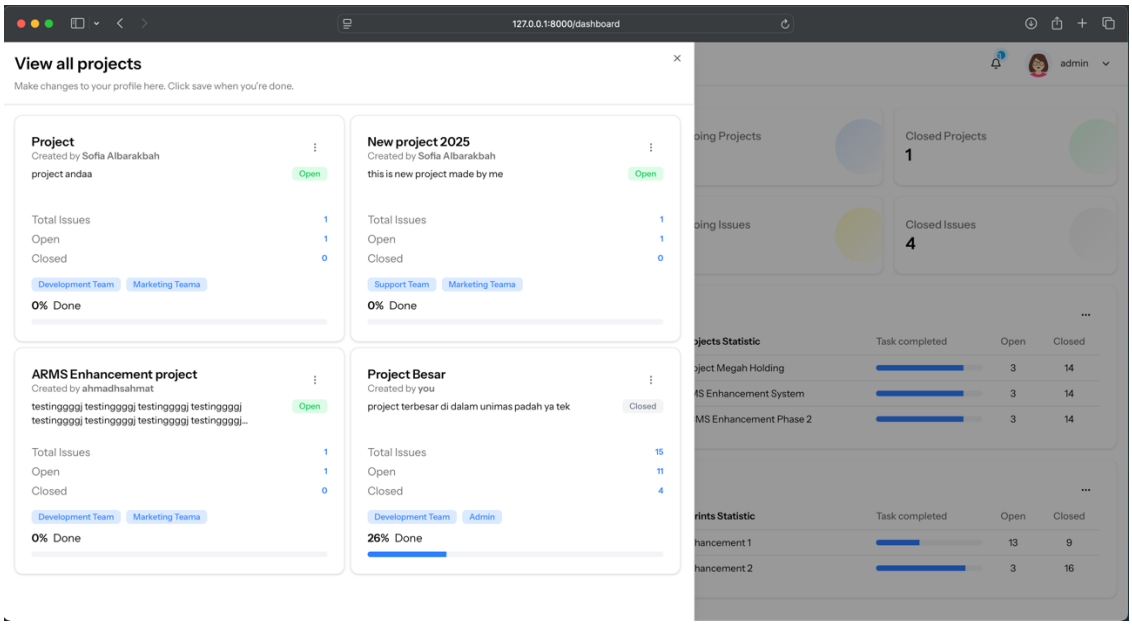



Figure 48: View all projects interface

#### 4.3.3.10 View, Update and Delete project

The View Project component displays comprehensive details of a selected project, including the project title, description, status, and the list of assigned teams. All team members associated with the project can update project details. However, the access to delete the project is restricted and can only be performed by the author (creator) of the project, ensuring proper control and accountability over project management actions. Figure 49 below shows the manage project user interface.

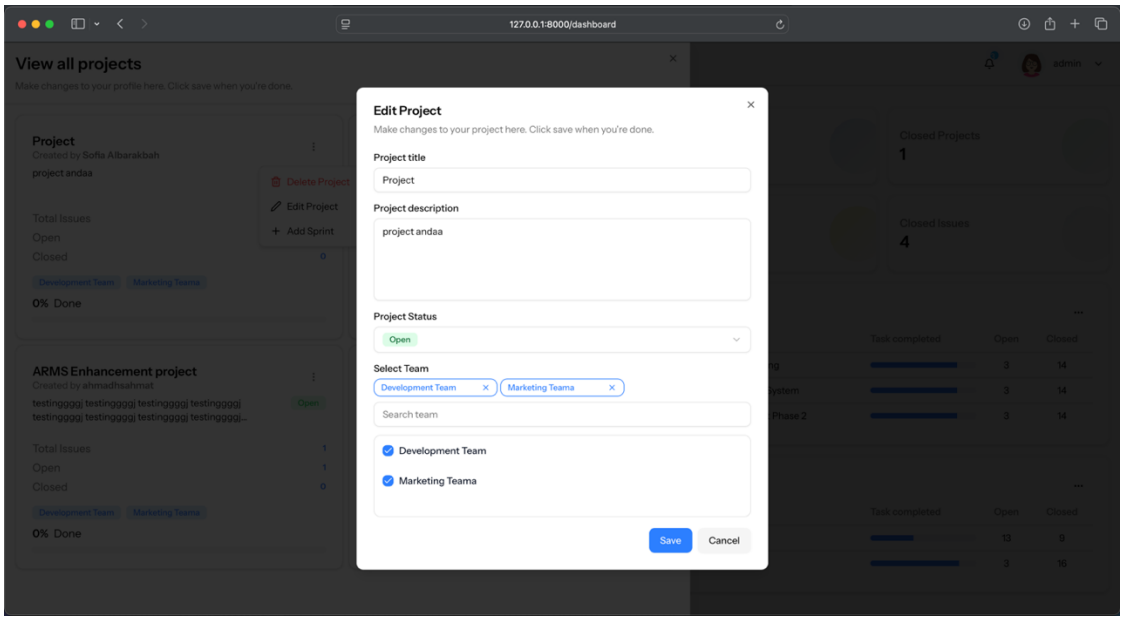



Figure 49: Manage project interface

#### 4.3.3.11 Create team

Users can create a new team by clicking the "+" button in the Teams section of the sidebar. This opens a form where users are required to fill in and select the necessary details to set up the team properly. These details including title, description, color label and users. Figure 50 below shows the create team user interface.

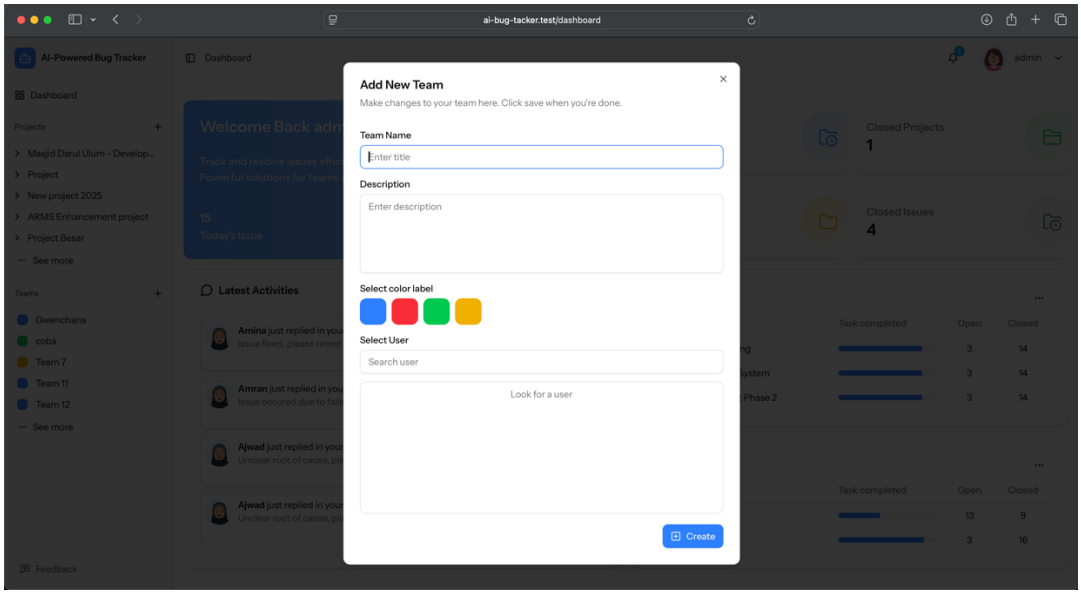



Figure 50: Create team interface

### 4.3.3.12 View all teams

In the View All Teams component, users can view a list of all teams that have been created and assigned within the system. Each team displays its details including the team's name, description, author (creator), and a list of team members. Figure 51 below shows the view all team user interface.

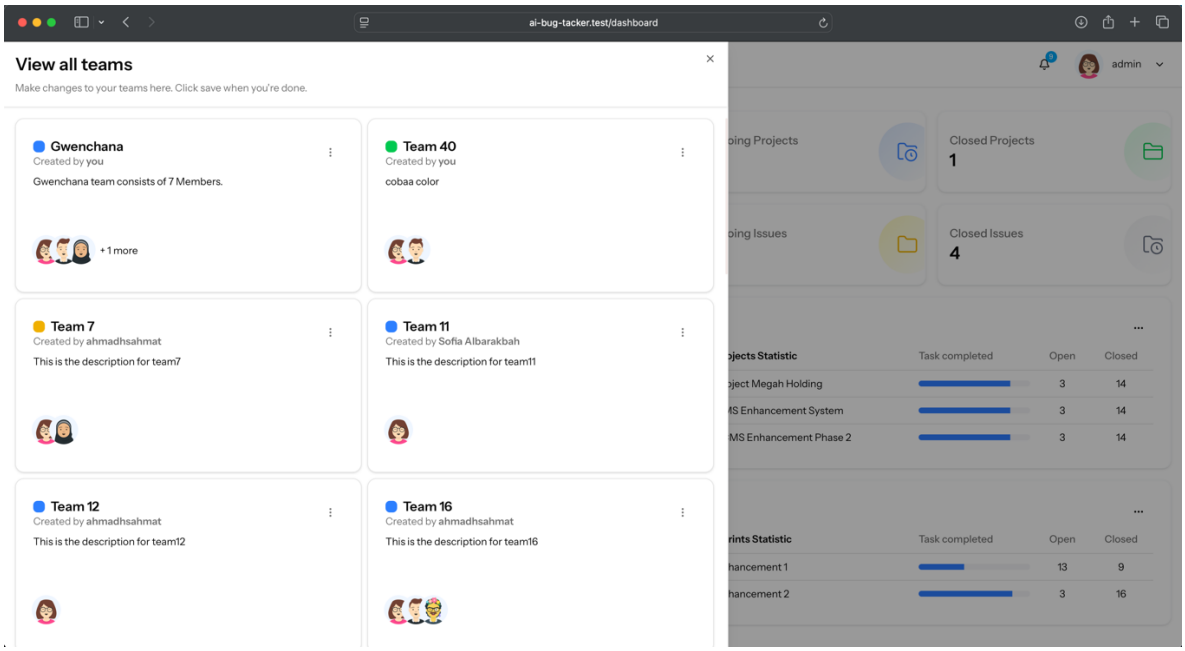



Figure 51: View all team interface

#### 4.3.3.13 View, Update and Delete/Leave team

The View Team component provides a detailed and editable view of a specific team, displaying information such as the team title, description, color label, and list of associated users. All users who are part of the team can update the team details. However, the access to delete the team is restricted to the team's author (creator) only. Users who are not the author will instead see a "Leave Team" option, allowing them to remove themselves from the team if needed. Figure 52 below shows the manage team user interface.

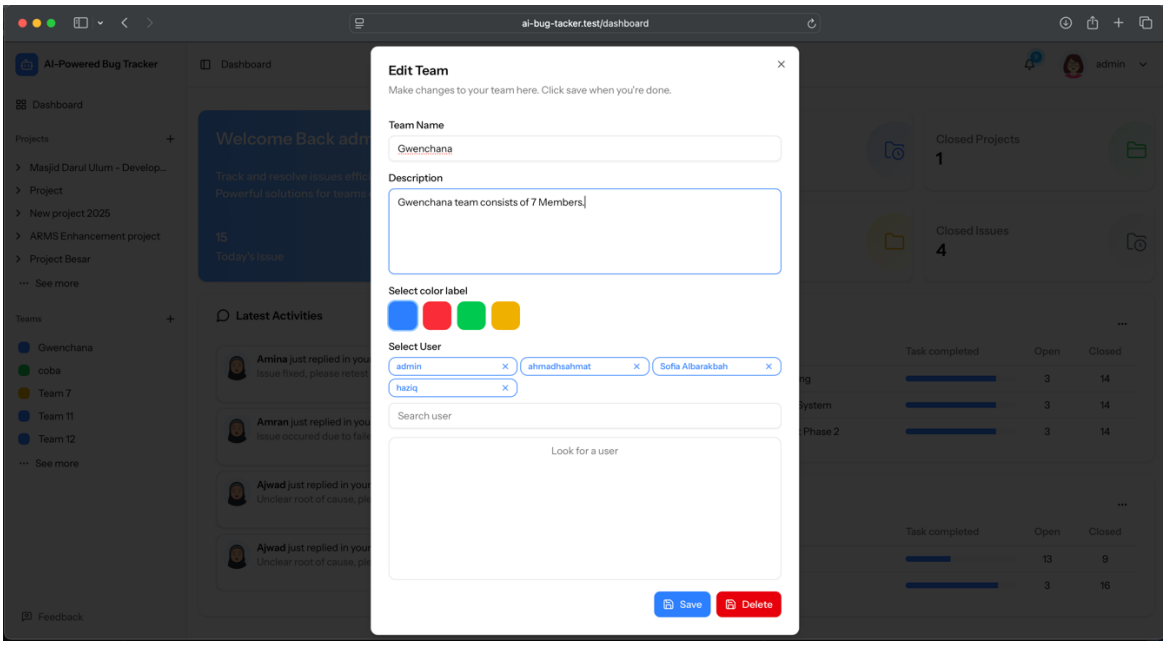



Figure 52: Manage team interface

#### 4.3.3.14 Project summary

Figure 53 shows the project summary user interface. In the View project summary module, users can view the statistic of each project with its sprints including:

1. Total Issues

2. Total of Open Issues
3. Total of Closed Issues
4. Total issue reported by each tester
5. Total issue by severity
6. Closed vs Open issues
7. Longest open issues
8. Sprint Summary

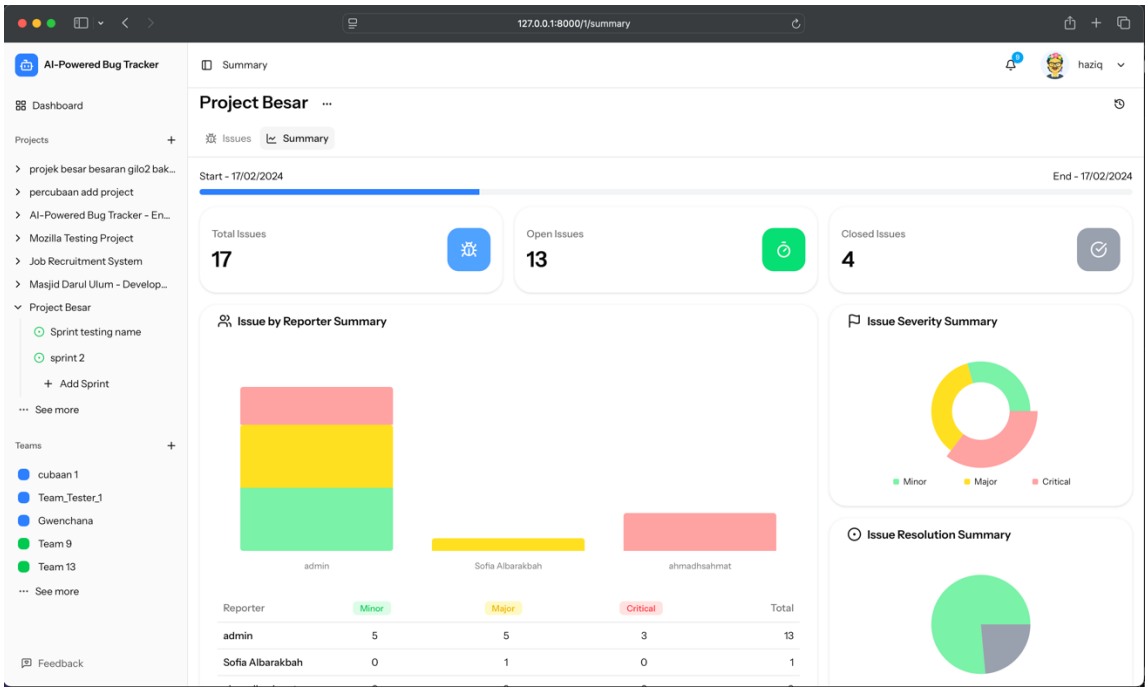



Figure 53: Project summary interface

## 4.4 Implementation of the AI features

This section explains how the AI features including bug severity prediction, Bug summarization and similar bug detection implemented in this project. Each feature will be described in detail, highlighting its features, and how it implemented and integrated into the system.

### 4.4.1 Bug Severity Prediction

The bug severity prediction feature is developed using a supervised machine learning approach. Specifically, the Multinomial Logistic Regression algorithm is employed to train the model on historical bug report data. Each report is represented using numerical features extracted from the text (such as the title and description), and the model learns to classify the severity level (e.g., critical, major, minor) based on these inputs.

#### 4.4.1.1 Dataset

Kaggle provide an open-source datasets and available for all users to use. In this project, a dataset titled “Bugzilla Bug Reports” will be used to train and test the machine learning model. This dataset consists of 50,000 bug reports from the Bugzilla project. It has been widely used by developers, researchers, and practitioners for various projects and research purposes.

#### 4.4.1.2 Data Cleaning and Preprocessing

The process starts with data cleaning, Irrelevant columns were removed, and missing values were handled. Text normalization steps such as lowercasing, stopword removal, and lemmatization

were applied to prepare the textual data. Figures 54 and figure 55 illustrate the process of data cleaning and natural language processing respectively.

```
[1]: import pandas as pd

[86]: df = pd.read_csv('sev.csv')
df2 = df.drop(columns=['Unnamed: 0.1', 'Unnamed: 0', 'Label'])
df2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49437 entries, 0 to 49436
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Description  32463 non-null  object
 1   Severity     32375 non-null  object
dtypes: object(2)
memory usage: 772.6+ KB

[88]: df3 = df2.dropna(subset=['Description', 'Severity'])
df3.info()

<class 'pandas.core.frame.DataFrame'>
Index: 32375 entries, 0 to 49436
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Description  32375 non-null  object
 1   Severity     32375 non-null  object
dtypes: object(2)
memory usage: 758.8+ KB
```

Figure 54: Data cleaning for severity prediction model

```
[104]: import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

*[106…: stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess(text):
    tokens = word_tokenize(text.lower())
    filtered_tokens = [lemmatizer.lemmatize(t, "v")
    for t in tokens if t.isalpha() and t not in stop_words]
    return filtered_tokens

df4['Description'].apply(preprocess)
```

[106]:

| | Description | Severity |
|---|---|---|
| 0 | Created by Darrell Kindred dkindred mozilla ... | minor |
| 1 | Created by Darrell Kindred dkindred mozilla ... | normal |
| 2 | Created by msuencks marcant de on Tuesday... | normal |
| 3 | Created by Darrell Kindred dkindred mozilla ... | major |
| 4 | Created by Darrell Kindred dkindred mozilla ... | trivial |
| ... | ... | ... |
| 32370 | Having a menu option to do an incremental ref... | normal |
| 32371 | Overview \nA document with an IMG whose SRC... | critical |
| 32372 | 2000052120 and older \nOpen Address Book or ... | normal |
| 32373 | Overview Description \n A minimal document se... | normal |
| 32374 | 2000052120\nrepeatedly click near the edge of... | normal |

32375 rows × 2 columns

Figure 55: Natural language processing for severity prediction model

#### 4.4.1.3 Vectorization

The cleaned bug descriptions were converted into numerical form using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique to capture the importance of

words across the dataset. Figure 56 illustrates the process of vectorization for severity prediction model.

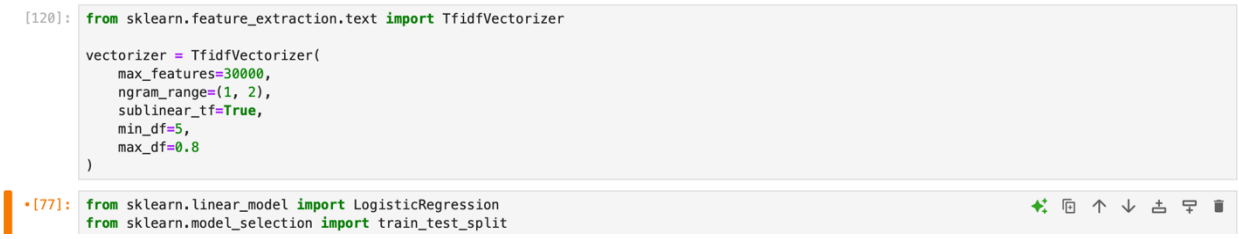



Figure 56: Vectorization for severity prediction model

#### 4.4.1.4 Model training and testing

A Multinomial Logistic Regression model was trained to classify the severity levels (e.g., Minor, Major, Critical) based on the vectorized text features. The dataset was split into training and testing sets to evaluate the model's generalizability. During this phase, it was observed that the dataset had an imbalance in the distribution of severity classes, which could lead to biased predictions. To address this, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied to the training data. This technique synthetically generates new instances of the minority class to balance the dataset, thereby improving the model's ability to learn from all classes without bias. Figure 57 shows the process of model training for the severity prediction model.

```
[77]: from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
X_text = df9['Description']  # text data (string)
y = df9['Severity']

X_vec = vectorizer.fit_transform(X_text)

X_train, X_test, y_train, y_test = train_test_split(
    X_vec, y, test_size=0.2, stratify=y, random_state=42
)

# 4. SMOTE only on training set
sm = SMOTE(random_state=42)
X_resampled, y_resampled = sm.fit_resample(X_train, y_train)

# 5. Train Logistic Regression
model_2 = LogisticRegression(
    random_state=42,
    max_iter=1000,
    multi_class='multinomial',
    solver='lbfgs'
)

model_2.fit(X_resampled, y_resampled)
```

Figure 57: Model training for severity prediction model

### 4.4.1.5 Model Evaluation

Evaluation metrics such as accuracy, precision, recall, and F1-score were calculated to assess the model's performance. Figure 58 shows the evaluation of the trained model.

```
# 6. Predict and Evaluate
y_pred = model_2.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    critical       0.64      0.65      0.64       761
       major       0.52      0.48      0.50       754
       minor       0.60      0.64      0.62       780

    accuracy                           0.59      2295
   macro avg       0.59      0.59      0.59      2295
weighted avg       0.59      0.59      0.59      2295
```

Figure 58: Model evaluation for severity prediction model

### 4.4.1.6 Deployment

After satisfactory evaluation, the trained model and vectorizer were saved using joblib for later integration into the Flask-based backend system of the bug tracking tool. Figure 59 shows the process of model deployment using joblib.

```
[79]: import joblib

      joblib.dump(model_2, 'prediction_model_ver2.pkl')
      joblib.dump(vectorizer, 'vectorizer_ver2.pkl')

[79]: ['vectorizer_ver2.pkl']
```

Figure 59: Model deployment for severity prediction model

Figure 60 shows the integration of the model and the system using flask. The system and the AI model will communicate with each other through API call.

```
49
50
      1 usage (1 dynamic)
51    @app.route( rule: "/prediction", methods=['POST'])
52    def predict():
53        data = request.json
54        raw_text = data.get('text', '')
55        if not raw_text:
56            return jsonify({'error': 'No text provided'}), 400
57        try:
58            X_vec = vectorizer.transform([raw_text])
59            prediction = model.predict(X_vec)
60            return jsonify({'prediction': prediction[0].capitalize()})
61        except Exception as e:
62            return jsonify({'error': str(e)}), 500
63
64    if __name__ == "__main__":
65        app.run(debug=True)
66
```

Figure 60: Model integration for severity prediction model

### 4.4.2 Bug Summarization

To implement the bug summarization feature, the Azure OpenAI service is used to generate concise and readable summaries of reported issues. The system integrates with the Azure OpenAI API by sending bug content along with a carefully designed prompt. The Gpt-4o model is specified in the API call to perform the summarization task, due to its strong language understanding and summarization capabilities. Figure 61 below shows the Gpt-4o model description.

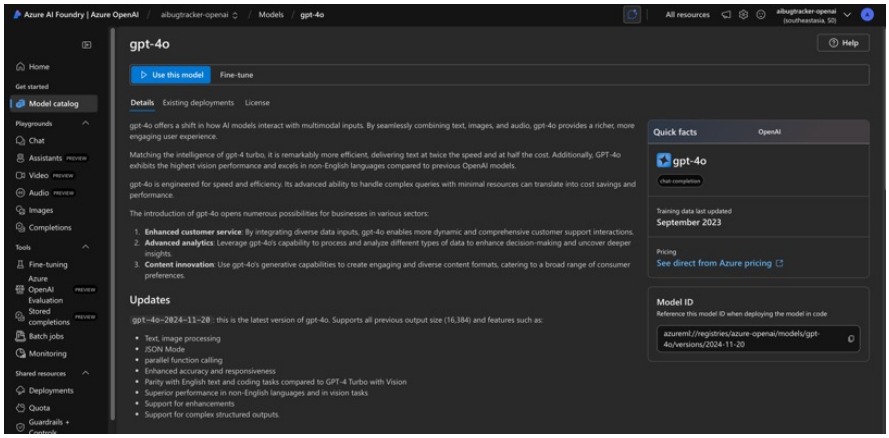



Figure 61: Gpt-4o Model selection from Azure AI foundry

The system and the OpenAI model will communicate with each other through API call. Figure 62 below shows the integration of the OpenAI and the system through API endpoint in OpenAIController.

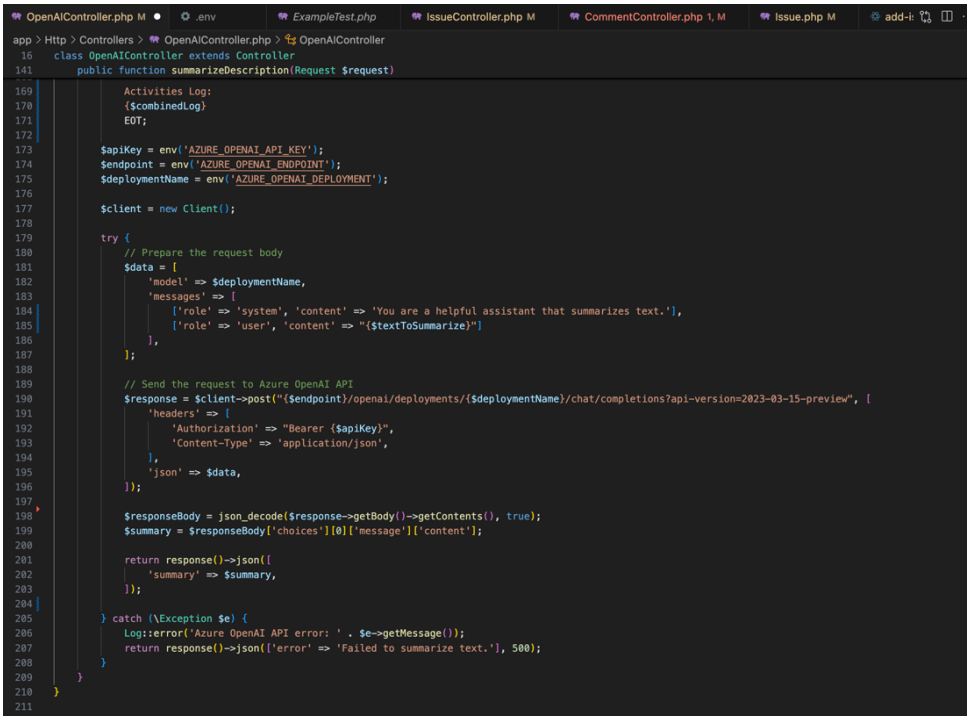



Figure 62: Azure service integration through API endpoint

### 4.4.3 Similar bug detection

The bug similarity prediction feature is designed to help detect duplicate or related bug reports by automatically identifying similar existing issues. This feature is implemented using an unsupervised machine learning approach based on text similarity.

#### 4.4.3.1 Vectorization

All bug descriptions are first transformed into numerical vectors using the TF-IDF (Term Frequency-Inverse Document Frequency) technique. This representation captures the importance of terms in each bug report relative to the entire corpus. Figure 63 shows the process of vectorization for bug similarity detection model.

```
[661]: from sklearn.feature_extraction.text import CountVectorizer
       vectorizer = CountVectorizer(stop_words='english')
       vector = vectorizer.fit_transform(df['tags'].values.astype('U')).toarray()
       df['tags']

[661]: 0    expose a small  simple ui for 'undo closed tab...
       1    undo close tab (a way to quickly reopen a clos...
       Name: tags, dtype: object
```

Figure 63: Vectorization for bug similarity detection model

#### 4.4.3.2 Similarity Computation

After vectorization, cosine similarity is used to compute the similarity scores between the newly submitted bug and all existing bug reports in the database. Cosine similarity measures the cosine of the angle between two vectors, giving a value between 0 (no similarity) and 1 (identical). Model deployment, Flask route calls model, returns similarity results). Figure 64 shows the process of similarity computation for bug similarity detection model.

```
[663]: from sklearn.metrics.pairwise import cosine_similarity
       similarity = cosine_similarity(vector)
       similarity

[663]: array([[1.        , 0.38692067],
              [0.38692067, 1.        ]])
```

Figure 64: Similarity computation for bug similarity detection model

#### 4.4.3.3 Integration and Optimization

The bug similarity feature is integrated into the backend of the system using the Flask web framework. A POST request is sent to an endpoint such as /api/similarity, containing the bug title and description in the request body. It loads the TF-IDF vectorizer and transforms the input text and all existing bug in a project into a vector. The system calculates similarity scores using cosine similarity and selects the top with score above 0.3. The result will be returned in JSON format for the frontend to render. Both components will communicate to each other using API call. Figure 65 shows the integration of the AI model and the system using flask for bug similarity detection feature.

```python
@app.route( rule: "/similarity", methods=['POST'])
def similarity():
    data_issues = request.json
    issues = data_issues.get("issues", [])
    df = pd.DataFrame(issues)
    df["tags"] = df["issueTitle"] + df["issueDesc"]
    target_input = pd.DataFrame([data_issues['target']])
    target_id = target_input.id[0]
    target_title = target_input.issueTitle[0]

    vectorizer = CountVectorizer(stop_words='english')
    vector = vectorizer.fit_transform(df['tags'].values.astype('U')).toarray()

    similarity = cosine_similarity(vector)
    index = df[df['issueTitle'] == target_title].index[0]
    distance = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda vector: vector[1])

    results = []

    for i in distance:
        if i[1] > 0.3:
            if df.iloc[i[0]].id != target_id:
                results.append(int(df.iloc[i[0]].id))

    return jsonify({"similarIssues": results})
```

Figure 65: Model integration for bug similarity detection

## 4.5 Summary

This chapter detailed the implementation of the AI-powered bug tracking system. It began by outlining the installation and configuration of the development environment and tools used throughout the project. The system's core features were implemented, including issues, project, team management, summary and AI functionalities. Three main AI components were then integrated, such as bug severity prediction, bug similarity detection and bug summarization.

# CHAPTER 5: SYSTEM TESTING

This chapter presents the testing process conducted to ensure that the AI-powered bug tracking system functions correctly, reliably, and meets the defined project requirements. Testing is a crucial phase in the software development lifecycle as it helps identify and resolve errors, validate features, and assess the system's usability under various conditions. This chapter also details the testing technique used, tools involved, and the results obtained, including user feedback on the system's usability.

## 5.1 Functional Testing

Functional testing was conducted to verify that each feature of the AI-powered bug tracking tool performs its intended function correctly. This type of testing concentrated on the core features of the system. The testing process used black-box testing technique, which focuses on evaluating the system's functionality without considering internal code structure.

### 5.1.1 Test Cases

Test cases are specific conditions or sets of inputs used to verify that a software application or system behaves as expected. Each test case is designed to check whether a particular feature or function works correctly under a defined scenario. The test steps and test data also provided to guide the testing process. The actual result will be compared with the expected result to determine whether the test case is passed or failed.

#### 5.1.1.1 System Testing

System testing will cover all the core functionalities or modules in the system. Each module will have its own test case, steps and scenario. This system testing will cover all modules including authentication and authorization (Table 28), user profile (Table 29), team management (Table 30), project management (Table 31), and issue management (Table 32).

Table 27: Test case for authentication & authorization

| Module Name: Authentication & Authorization<br>Test Case Description: This test case is to test the login, registration and forgot password features | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-AA-1 | User should be able to login with valid credentials | 1. Fill in valid email address<br>2. Fill in valid password<br>3. Click "login" button | Email: ahmad@gmail.com<br><br>Password: ahmad123 | Expected: User navigate to dashboard<br>Actual: User navigate to dashboard successfully | Passed |
| TC-AA-2 | User should not be able to login with invalid credentials | 1. Fill in invalid email address<br>2. Fill in invalid password<br>3. Click "login" button | Email: ahad@gmail.com<br><br>Password: ahmad12 | Expected: User navigate to login page<br>Actual: User navigate to login page and error message appeared | Passed |
| TC-AA-3 | User should be able to register account | 1. Fill in valid user details<br>2. Click "register" button | Email: testaccount@gmail.com<br><br>Username: testaccount1<br><br>Password: testaccount1<br><br>Role: Developer | Expected: User successfully registered and navigate to their dashboard<br>Actual: User account successfully registered and their dashboard shown | Passed |

| | | | | | |
|---|---|---|---|---|---|
| TC-AA-4 | User should not be able to register account with invalid data | 1. Fill in invalid user details<br>2. Click “register” button | Email: testaccount22gmailcom<br><br>Username: 0191920939304<br><br>Password: Testaccoun t44<br><br>Role: Undefined | Expected: User account registration is failed<br>Actual: User account registration is failed | Passed |
| TC-AA-5 | User should be able to reset their password | 1. Fill in valid user email<br>2. Click “email password reset link” button | Email: testaccount@gmail.com | Expected: Reset password link will be send through email<br>Actual: Reset password link will be send through email | Passed |

Table 28: Test case for user profile

| Module Name: User Profile<br>Test Case Description: This test case is to test the user profile features | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-UP-1 | User should be able to update their details | 1. Navigate to user settings<br>2. Update user details with valid data<br>3. Click save button | Email: testaccount2@gmail.com<br><br>Username: testaccount2<br><br>Password: testaccount2<br><br>Role: Business Analyst | Expected: User details successfully updated<br>Actual: User details successfully updated | Passed |
| TC-UP-2 | User should not be able to update their details with invalid data | 1. Navigate to user settings<br>2. Click Profile tab<br>2. Update user details | Email: testacnt200gmailco<br><br>Username: 019283<br><br>Role: | Expected: User details update failed<br>Actual: User details failed to update | Passed |

| | | | | | |
|---|---|---|---|---|---|
| | | with invalid data<br>3. Click save button | Undefined | | |
| TC-UP-3 | User should be able to update their password | 1. Navigate to user settings<br>2. Click password tab<br>3. Update user password with valid data<br>4. Click save button | Current password: testaccount2<br><br>New password: testaccount234<br><br>Confirm password: testaccount234 | Expected: User password updated successfully<br>Actual: User password updated successfully and able to login with new password | Passed |
| TC-UP-4 | User should not be able to update their password with invalid data | 1. Navigate to user settings<br>2. Click password tab<br>3. Update user password with invalid data<br>4. Click save button | Current password: testaccount2<br><br>New password: 00 Test account 234<br><br>Confirm password: 00 Test account 234 | Expected: User password update failed<br>Actual: User password update failed | Passed |
| TC-UP-5 | User should be able to delete their account | 1. Navigate to user settings<br>2. Click Profile tab<br>3. Click “Delete Account” button | Password: testaccount234 | Expected: User account will be deleted and redirect to login page<br>Actual: User account will be deleted and redirect to login page | Passed |

Table 29: Test case for team management

| Module Name: Team management<br>Test Case Description: This test case is to test the team management module with valid and invalid data. | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-TM-1 | User should be able create team | 1. Fill in team details<br>2. Click “create” button | Name: Team test 1<br><br>Description: Team description for test data<br><br>Color Label: Blue | Expected: Team created successfully<br>Actual: Team created successfully | Passed |
| TC-TM-2 | User should be able to assign team members | 1. Search users<br>2. Select team member | Users: testaccount5, testaccount6, testaccount7 | Expected: Users successfully assigned to the team<br>Actual: Users successfully assigned to the team | Passed |
| TC-TM-3 | User should not be able add team with invalid data | 1. Fill in invalid team details<br>3. Click “create” button | Name: 0<br>0393992@@<br><br>Description: 0<br><h1> project description </h1> !!nd<br><br>Color Label: Undefined/Empty field | Expected: Error message shown in each input field<br>Actual: Error message shown in each input field | Passed |
| TC-TM-4 | User should be able to delete team | 1. View team to delete<br>2. Click “Delete” button | Team Name: Team test 1 | Expected: Team deleted successfully<br>Actual: | Passed |
| TC-TM-5 | User should be able to leave team | 1. View team to leave<br>2. Click “Leave” button | Team Name: Team test 1 | Expected: User successfully leave team<br>Actual: | Passed |

| | | | | | |
|---|---|---|---|---|---|
| | | | | User successfully leave team | |
| TC-TM-6 | User should be able to update team details | 1. View team to edit<br>2. Change team details<br>3. Click “Update” button | Name: Team 89<br><br>Description: Team description for update data<br><br>Color Label: Yellow | Expected:<br>Team updated successfully<br><br>Actual:<br>Team updated successfully | Passed |
| TC-TM-7 | User should not be able to update team details with invalid data | 1. View team to edit<br>2. Change team with invalid details<br>3. Click “Update” button | Name: <br><h1> team name </h1> !!nd<br><br>Description: U<br><h1> team description </h1><br><br>Color Label: Undefined | Expected:<br>Team update failed<br><br>Actual:<br>Team update failed | Passed |

Table 30: Test case for project management

| Module Name: Project management<br>Test Case Description: This test case is to test the team management module with valid and invalid data. | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-PM-1 | User should be able to create project | 1. Fill in project details<br>2. Click “Create” button | Name: Project Test 1<br><br>Description: Project description for test data | Expected: Project created successfully<br>Actual: Project created successfully | Passed |
| TC-PM-2 | User should be able to assign team | 1. Search team<br>2. Select teams | Teams:<br>Team test 1,<br>Team test 2 | Expected: Teams successfully assigned to the project<br>Actual: Teams successfully assigned to the project | Passed |

| | | | | | |
|---|---|---|---|---|---|
| TC-PM-3 | User should not be able add project with invalid data | 1. Fill in invalid project details<br>3. Click “create” button | Name: 0<br>0393992@@<br><br>Description: ---00393992@@  description  | Expected: Error message shown in each input field<br>Actual: Error message shown in each input field | Passed |
| TC-PM-4 | User should be able to delete project | 1. View project to delete<br>2. Click “Delete” button | Name: Project Test 1 | Expected: Project deleted successfully and user will be redirected to dashboard<br>Actual: Project deleted successfully, and user will be redirected to dashboard | Passed |
| TC-PM-5 | User should be able to update project details | 1. View project to edit<br>2. Change project details with valid data<br>3. Click “Update” button | Name: Project Test 12<br><br>Description: Project Test 12 description for test data | Expected: Project updated successfully<br>Actual: Project updated successfully | Passed |
| TC-PM-6 | User should not be able to update project details with invalid data | 1. View project to edit<br>2. Change project details with invalid data<br>3. Click “Update” button | Name: <h1 > Project Name  ! description <br><br>Description: ---00393992@@  description  | Expected: Project update failed<br>Actual: Project update failed | Passed |

Table 31: Test case for project summary

| Module Name: Project summary<br>Test Case Description: This test case is to test the project and its sprint summary | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-PS-1 | User should be able to see project summary | 1. Select any project<br>2.Click summary tab | N/A | Expected: Project created successfully<br>Actual: Project created successfully | Passed |
| TC-PS-2 | All summaries details are correct | 1. Click any project<br>2. Click summary tab | N/A | Expected: Project showed correct details<br>Actual: Project showed correct details | Passed |

Table 32: Test case for issue management

| Module Name: Issue management<br>Test Case Description: This test case is to test the issue management module with valid and invalid data. | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-IM-1 | User should be able add issue with valid data | 1. Fill in issue details<br>2. Click “submit” button | Title: Unable to click add button<br><br>Description: In project module, the add button is disabled,<br><br>Assignee: test_account4<br><br>Due Date: 13/8/2025<br><br>Severity: Critical<br><br>Root of cause: Incorrect form validation | Expected: Issue submitted successfully<br>Actual: Issue submitted successfully and shown on the list | Passed |

| | | | | | |
|---|---|---|---|---|---|
| | | | Activities:<br>Screenshot is attached below, test.png | | |
| TC-IM-2 | User should not be able add issue with invalid data | 1. Fill in invalid bug details<br>3. Click “submit” button | Title: <br>< h1 > Project && Name!!!</h1><br><br>Description: <br>< h1 > == Project Name ! description <br><br>Assignee:<br>test_account4<br><br>Due Date:<br>40/13/0003<br><br>Severity:<br>Critical<br>Root of cause:<br>Undefined/Empty field<br><br>Activities:<br>Undefined/Empty field | Expected: Error message shown in each input field<br><br>Actual: Error message shown in each input field | Passed |
| TC-IM-3 | User should be able to delete issue | 1. View issue to delete<br>2. Click “Delete” button | N/A | Expected:<br>Issue deleted successfully<br><br>Actual: Issue deleted successfully | Passed |
| TC-IM-4 | User should be able to update issue with valid data | 1. View issue to edit<br>2. Change issue details<br>3. Click “Update” button | Title: Unable to locate delete button<br><br>Description: In project module, the delete button is missing<br><br>Assignee:<br>test_account3 | Expected:<br>Issue updated successfully<br><br>Actual:<br>Issue updated successfully | Passed |

| | | | | | |
|---|---|---|---|---|---|
| | | | Due Date:<br>14/8/2025<br><br>Severity:<br>Major<br>Root of cause:<br>Incorrect source code | | |
| TC-IM-5 | User should not be able to update issue with invalid data | 1. View issue to edit<br>2. Change issue details with invalid data<br>3. Click “Update” button | Title: <br>< h1 > Project Name  ! !!&&<br><br>Description: <br>< h1 > Project && description  ! description <br><br>Assignee:<br>test_account3<br><br>Due Date:<br>39/14/0002<br><br>Severity:<br>Undefined/Empty field<br><br>Root of cause:<br>Undefined/Empty field | Expected:<br>Issue update failed<br><br>Actual: Issue update failed | Passed |
| TC-IM-6 | User should be able to add comment in the issue | 1. Click any issue to view<br>2. Add comment in the activities section | Comment: This is to test comment input field | Expected:<br>Comment will be sent successfully<br><br>Actual: Comment will be sent successfully | Passed |
| TC-IM-7 | User should not be able to add invalid comment in the issue | 1. Click any issue to view<br>2. Add comment in the activities section with invalid data | Comment: <h1><br><br>T@#his is updated @^&jjdkkff</h1> | Expected:<br>Error message shown in each input field<br><br>Actual: Comment Error message | Passed |

| | | | | | |
|---|---|---|---|---|---|
| | | | | shown in each input field | |
| TC-IM-8 | User should be able to update their comment | 1. Click any issue to view<br>2. Click action button on any comment and click “edit” button<br>3. Click “Save” button | Comment: This is updated comment and different from previous one | Expected: Comment updated successfully<br>Actual: Comment updated successfully with correct result | Passed |
| TC-IM-9 | User should not be able to update their comment with invalid data | 1. Click any issue to view<br>2. Click action button on any comment and click “edit” button<br>3. Click “Save” button | Comment: <h1> <br>T@#his is upded @^&jjdkkff</h1> | Expected: Comment update failed<br>Actual: Comment update failed | Passed |
| TC-IM-10 | User should be able to delete comments in the issue | 1. Click any issue to view<br>2. Click action button on any comment and click “delete” button | N/A | Expected: Comment deleted successfully<br>Actual: Comment deleted successfully | Passed |
| TC-IM-11 | User should be able add attachment in their comment | 1. Click any issue to view<br>2. Click “Add attachment” button<br>3. Insert attachment<br>4. Click “Send” button | File format allowed: png, jpeg, jpg | Expected: Attachment sent successfully<br>Actual: Attachment sent successfully | Passed |

#### 5.1.1.2 Integration Testing

Integration testing was performed to ensure that the system's components work together as intended, particularly focusing on the integration of the AI features (Table 34) with the core system. The system includes three key AI-powered features including bug severity prediction, similarity detection and summarization. These AI modules are integrated into the system via API communication. The integration was tested using the Postman tool to simulate API requests and validate the communication between the system and the AI services. Inputs were sent to each API endpoint, and the outputs were verified to ensure they were accurate, consistent, and aligned with the expected results.

Table 33: Test case for AI & Machine Learning integration

| Module Name: AI and machine learning integration<br>Test Case Description: To test three key AI-powered features including bug severity prediction, similarity detection and summarization | | | | | |
|---|---|---|---|---|---|
| **ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Result** | **Passed/Fail** |
| TC-AI-1 | User should be able predict bug severity | 1. Fill in bug description<br>2. Click "Ai-prediction" button | Description: Unable to view the Project module. Error message shown when tried to access the module. | Expected: The model able to produce severity<br>Actual: The model able to produce output (severity) after sending the bug description | Passed |
| TC-AI-2 | User should be able to summarize issue | 1. View any issue<br>2. Clicks summarize issue | N/A | Expected: The summary of an issue shown<br>Actual: The summary of an issue shown with relevant details | Passed |
| TC-AI-3 | User should be able to view similar issues | 1. Click "Add issue" button<br>2.Fill in bug details<br>3.View bug details<br>3. Navigate to similar reported issue section | Title: Unable to access project module<br>Description: Unable to view the Project module. Error message shown when tried to access the module. | Expected: The machine learning model able to receive and produce output<br>Actual: The machine learning model able to receive and produce relevant outputs | Passed |

## 5.2 Non-Functional Testing

This section focuses on non-functional testing, which assesses the quality attributes beyond its functional requirements. Non-functional testing ensures the system meets expectations in areas such as usability, reliability, performance, scalability, and maintainability. For this project, the non-

functional testing primarily emphasizes usability, evaluating how effectively and efficiently users can interact with the system.

### 5.2.1 Usability Testing

Usability testing was conducted to assess the overall user experience of the system. 10 participants from relevant fields were selected to perform a usability testing on the system. During the testing process, participants were observed as they interacted with the system to identify any usability challenges or confusion. At the end of the testing, they were instructed to evaluate and assess the system.

#### 5.2.1.1 Participants background

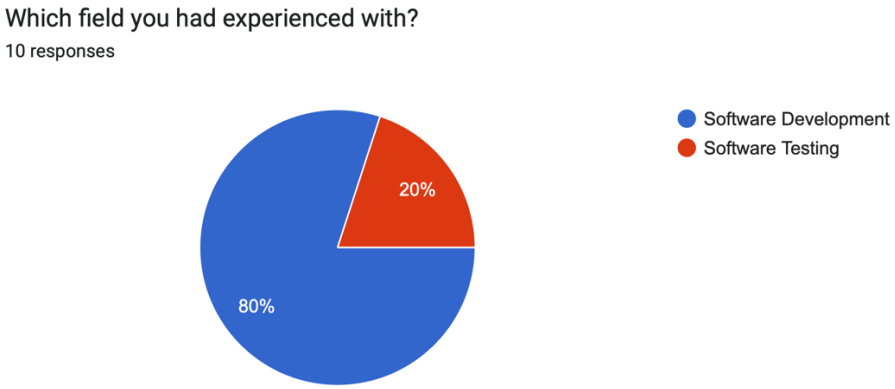



Figure 66: Results of participant's experience

The usability testing of the system was conducted with 10 participants from diverse backgrounds, including software development and testing. Figure 66 shows the percentage of the participant's experience.

#### 5.2.1.2 Ease of use

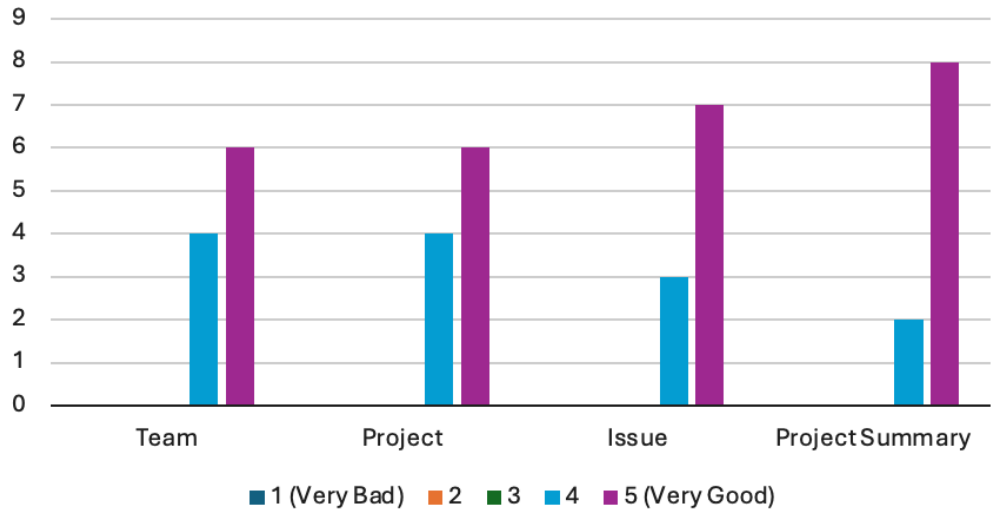



Figure 67: Results of the ease of use for each core module

Table 34: Results of the ease of use for each core module

| Module | 1 (Very Bad) | 2 (Bad) | 3 (Average) | 4 (Good) | 5 (Very Good) |
|---|---|---|---|---|---|
| Team Management | 0 | 0 | 0 | 4 | 6 |
| Project Management | 0 | 0 | 0 | 4 | 6 |
| Issue Management | 0 | 0 | 0 | 3 | 7 |
| Project Summary | 0 | 0 | 0 | 2 | 8 |

Based on figure 67, most modules including Team, Project, Issue, and Summary received scores between 4 (Good) and 5 (Very good) from the users. This indicates that most of the modules are perceived as very easy to use. These results support the system's usability goal, demonstrating

that users can interact with the tool efficiently and with minimal learning effort. Although the average score for the ease of use is between 4 (good) and 5 (very good), some users also believed that there is confusion when completing their tasks in specific modules.

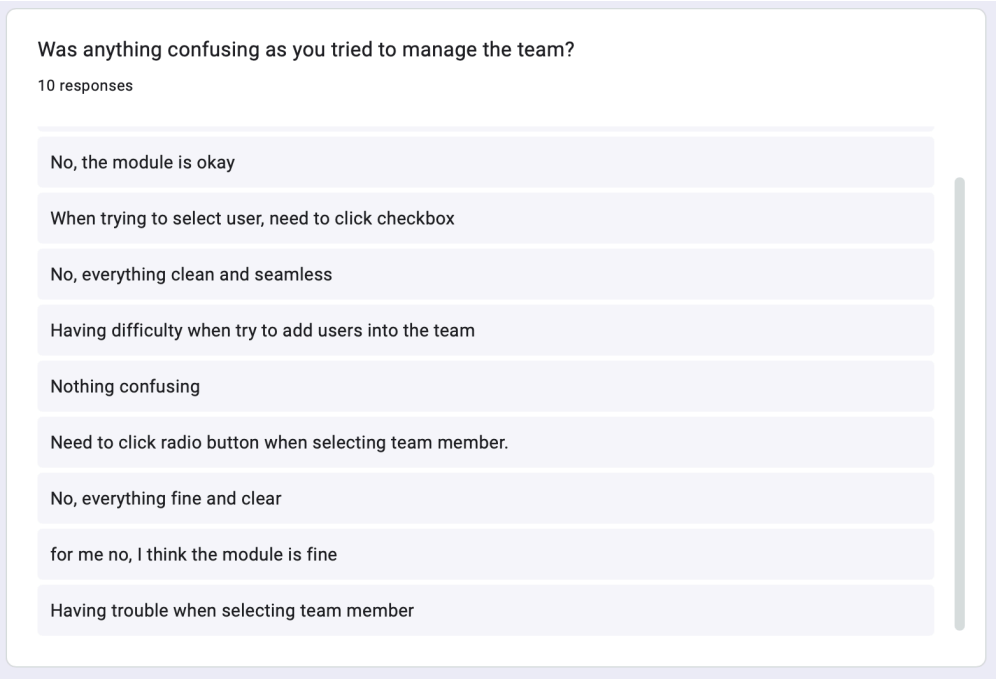



Figure 68: Results of user confusion in the team module

Based on figure 68, majority of the users believed the team module did not confuse them. However, some of the users found certain parts of module is confusing while performing their tasks. The main issue occurred when users attempted to select a team member, as they were unsure whether to click the card or the radio button, leading to confusion during the selection process.

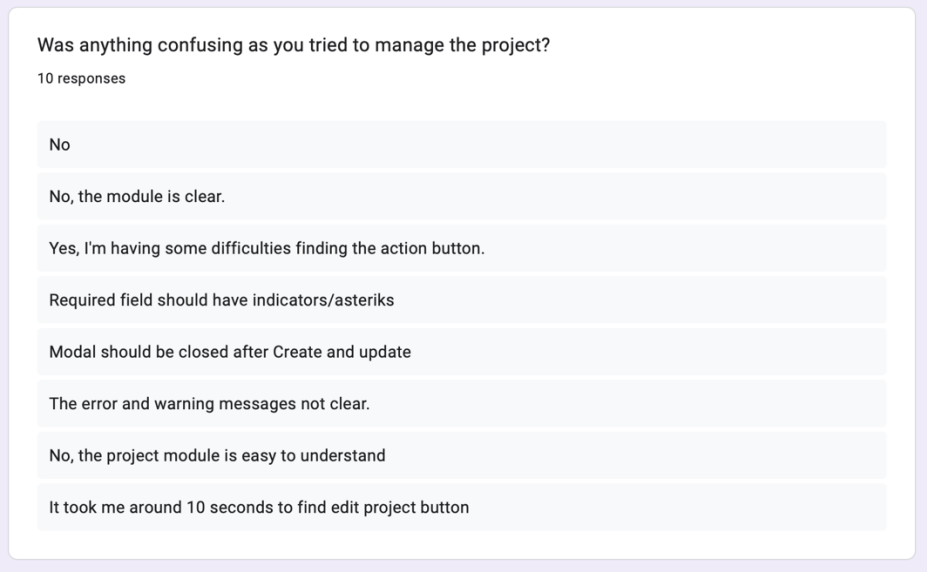


Figure 69: Results of user confusion in the project module

In figure 69, majority of the users found nothing confusing in project module. Some of the users found certain parts of the project module confusing while performing specific tasks. Most of the issues were related to the user interface (UI), where the system failed to communicate effectively with the user such as bad “action” button placement, missing asterisk for required field, improper modal flow, and unclear error message.

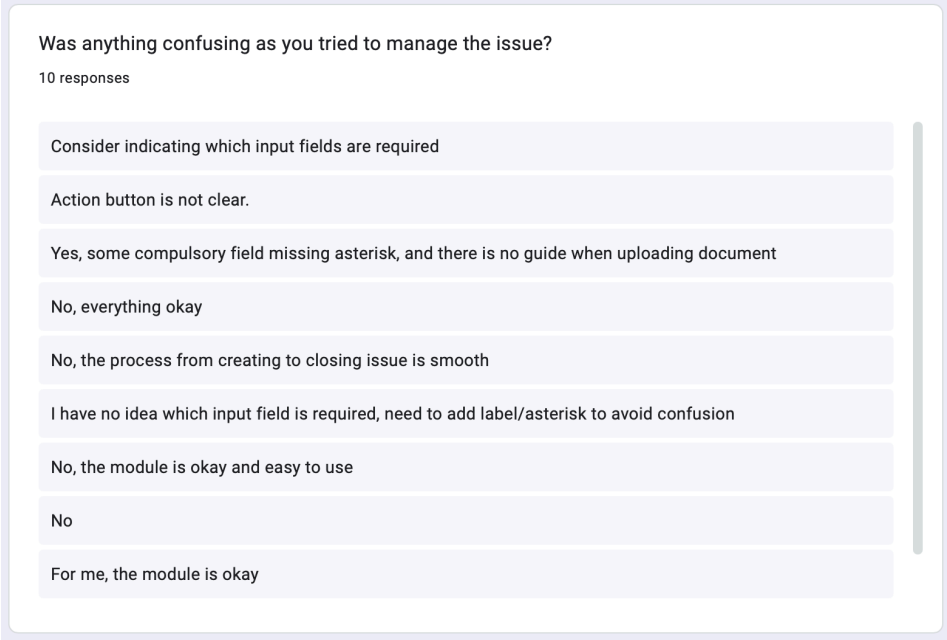



Figure 70: Results of user confusion in the issues module

Based on figure 70, some users found certain parts confusing when performing specific tasks in issue module. Users also found it difficult to look for “action” button and the form did not label each required field with asterisk. This cause user to spend a quite some time to look for action button as it is not at the desired area.

Although the users scored 4-5 for each module in term of ease of use, the confusion and challenges when completing the tasks still exists. Most of the issues were related to the user interface (UI), where the system failed to communicate with the users effectively. Hence, this confusion and challenges need to be eliminated and improve in order to increase the users satisfaction and experience when using the system.

### 5.2.1.3 Overall system

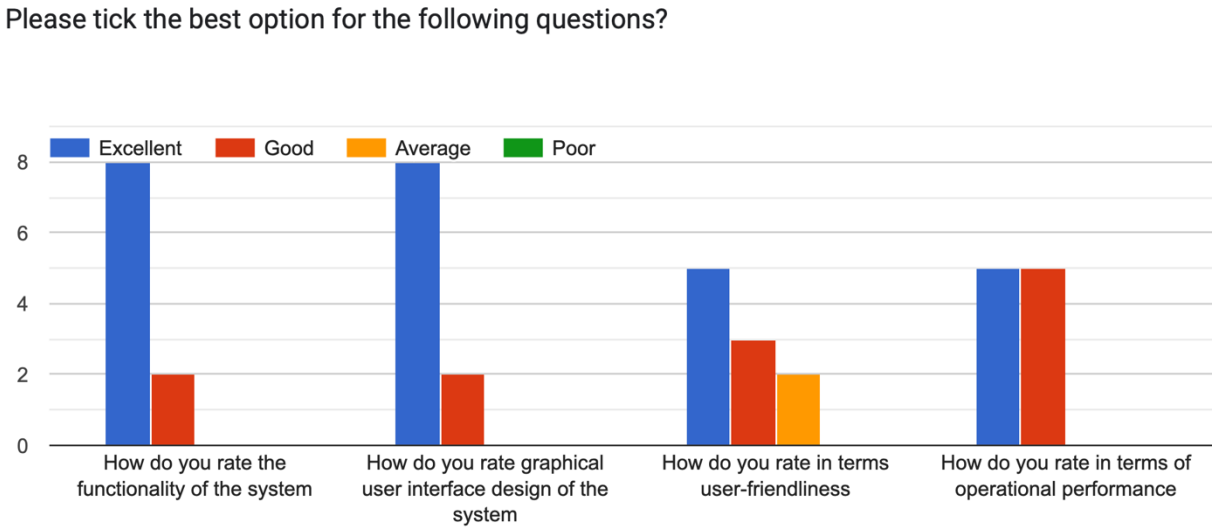



Figure 71: Results for overall system

Table 35: Results for overall system

| Questions | Poor | Average | Good | Excellent |
|---|---|---|---|---|
| How do you rate the functionality of the system | 0 | 0 | 2 | 8 |
| How do you rate graphical user interface design of the system | 0 | 0 | 2 | 8 |
| How do you rate in terms user-friendliness | 0 | 2 | 3 | 5 |
| How do you rate in terms of operational performance | 0 | 0 | 5 | 5 |

Based on Figure 71 and Table 35, the system received generally positive ratings across all usability dimensions. 8 out of 10 users rated the system's functionality as 'Excellent', while the other two rated it as 'Good', suggesting that core features worked as expected and aligned well

with user needs. The graphical user interface (GUI) was also well received, with 8 users selecting ‘Excellent’ and 2 selecting ‘Good’, indicating that the visual design and layout were clear and user-friendly.

User-friendliness, however, received slightly more mixed feedback: 5 users selecting ‘Excellent’, 3 selecting ‘Good’, and 2 selecting ‘Average’. The user who rated it as ‘Average’ expressed difficulty navigating certain modules, particularly when interacting with the team member selection interface and difficulty in finding the “action” button. This suggests that while the system is generally easy to use, some components may require improved guidance to support a wider range of user expectations.

Operational performance was rated as ‘Good’ by 5 and ‘Excellent’ by 5, indicating that the system functioned reliably during use. However, the absence of full marks from all users may suggest minor delays or uncertainty about how fast the system should respond, especially during API interactions with the AI modules.

Hence, while the usability profile is strong with the majority of feedback between ‘Good’ and ‘Excellent’ the few non-optimal ratings reveal specific areas that could benefit from refinement. These include enhancing UI intuitiveness in certain modules and improving feedback mechanisms to make system responses more transparent and reassuring.

### 5.2.1.4 AI and ML Integration

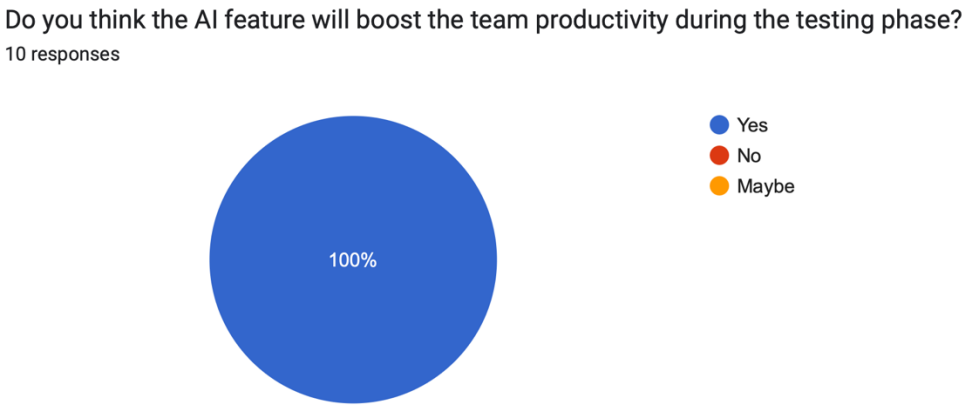



Figure 72: Results for AI features

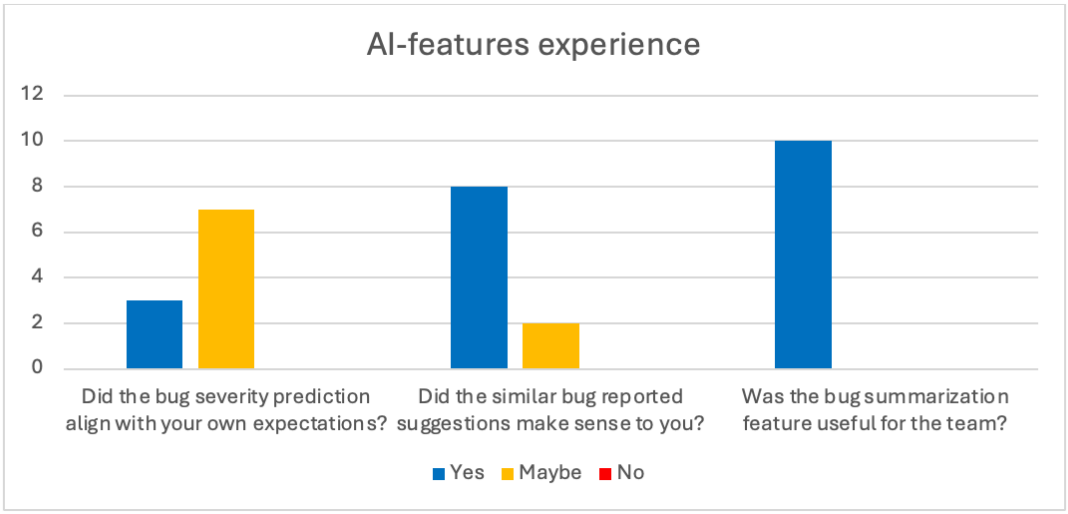



Figure 73: Results of user experience with the AI features

Based on Figure 72, 100% of the users agreed that the integrated AI features, such as bug severity prediction, summarization, and similar bug detection would enhance team productivity during the software testing phase. While users acknowledged the potential of these features, some expressed hesitation about fully relying on them. According to Figure 73, 7 out of 10 users were

unsure about the accuracy of the bug severity prediction. This uncertainty was mainly due to a lack of transparency in how the severity was determined, and in some cases, the AI prediction may not align with the users' own assessment and criteria based on the bug description. This suggests that the model may require better tuning or explanation of its reasoning to gain user trust.

In contrast, 8 out of 10 users expressed confidence in the similar bug detection feature, likely because it provided visible and relevant results that aligned with their expectations, making it easier for users to evaluate its accuracy. Additionally, all users were highly confident in the bug summarization feature, viewing it as the most helpful AI features in the system. Users appreciated how the summaries helped them quickly understand the context of a bug without reading the entire things.

Overall, while the integration of AI features significantly improves productivity and streamlines bug management, the findings indicate that features like severity prediction and similarity detection still require human intervention. Future improvements should focus on enhancing model accuracy and user understanding to increase confidence and reliance on AI-driven results.

## 5.3 Summary

In conclusion, different types of testing were carried out to make sure the system works correctly and is easy to use. System testing showed that all modules function as expected. Integration testing confirmed that the main features, including the AI components, work well together. Usability testing with users from different backgrounds showed that the system is user-friendly and useful for real-world software projects. Overall, the results show that the system meets its goals and is ready to be used or improved further.

# CHAPTER 6: CONCLUSION AND FUTURE WORKS

## 6.1 Introduction

This chapter presents the conclusion and outlines potential future work for the project. It provides a summary of the overall investigation, reiterating the objectives, and achievements discussed throughout the report. The section also reflects on the lessons learned during the development and evaluation of the AI-powered bug tracking tool, supported by relevant findings from testing and user feedback. Finally, suggestions for future enhancements and possible extensions are proposed to guide continued improvement and research.

## 6.2 Objectives and Achievements

The main objective of this project was to develop an AI-powered bug tracking tool that could assist software teams in managing bug reports more efficiently. Table below shows the objectives and its achievement.

Table 36: Objectives and achievements of the project

| Objectives | Achievements |
|---|---|
| To develop a bug tracking tool that able to keep track of software bugs during testing phase. | A fully functional bug tracking tool was successfully developed, enabling efficient tracking and management of software bugs throughout the testing phase. |

| | |
|---|---|
| To accelerate bug reporting process through the integration of Artificial Intelligence (AI) technologies. | The integration of AI technologies streamlined the bug reporting process, significantly reducing the time required to predict severity of bug. |
| To integrate Machine Learning (ML) subset of AI that learns bugs pattern and predict severity. | A Machine Learning model was effectively implemented to analyze historical bug data, enabling the system to learn patterns and predict the severity of software bug. |
| To evaluate the effectiveness of the proposed bug tracking tool for testers and developers during testing. | The effectiveness of the bug tracking tool was successfully evaluated through usability testing and software expert feedback, demonstrating improved efficiency and usability for both testers and developers. |

## 6.3 Limitations

The limitation of project is the severity levels can be subjective and vary between organizations or projects. This may affect the consistency of predictions, especially in edge cases. Other than that, the UI/UX inconsistencies need to be eliminated and improved as some users experienced confusion in specific modules (e.g., team member selection, missing asterisk), indicating that parts of the interface require improvement for better usability. Finally, most modern tool nowadays able to integrate with another systems. The tool operates independently and may

require additional integration work to be adopted into real-world development environments like Jira or GitHub Issues.

## 6.4 Future works

For future work, the system should support customizable severity labels and criteria, allowing organizations to tailor the prediction model to their specific workflow. The user interface should also be refined based on usability testing feedback such as improving the intuitiveness of team member selection and including guidance for new users. Additionally, the system should be extended to integrate with other popular issue trackers like Jira, GitHub Issues, and GitLab, enabling seamless bug reporting and analysis within existing development pipelines.

## 6.5 Conclusion

This project successfully developed an AI-powered bug tracking tool that enhances the bug management process through automated severity prediction, similarity detection, and bug report summarization. The integration of machine learning, natural language processing, and OpenAI technologies has shown significant potential in improving efficiency and reducing manual effort in software testing. Usability testing indicated that the system was generally well-received, though some user interface improvements are still needed. The project has achieved its core objectives and established a solid foundation for future enhancements. Overall, this work highlights the practical value of incorporating AI into software quality assurance processes and opens opportunities for further research and development in intelligent debugging systems.

## REFERENCES

Lethbridge, T. C., & Laganière, R. (2004). *Object-oriented software engineering: Practical software development using UML and Java*. McGraw-Hill College.

Otoom, A. F., Al-Shdaifat, D., Hammad, M., & Abdallah, E. E. (2016). Severity prediction of software bugs. *2016 7th International Conference on Information and Communication Systems (ICICS)*. https://doi.org/10.1109/iacs.2016.7476092

Hamouri, S. K., Shatnawi, R. A., AlZoubi, O., Migdady, A., & Yassein, M. B. (2023). Predicting bug severity using machine learning and ensemble learning techniques. 2023 14th International Conference on Information and Communication Systems (ICICS), 1-6. https://doi.org/10.1109/icics60529.2023.10330494

# APPENDIX A: PRE-SURVEY INTERVIEW QUESTIONS

Part 1: Software Testing

In this section, we aim to evaluate your experience with software testing practices. Your responses will help us understand how testing is approached within your team and, the tools you use.

*Dalam bahagian ini, kami bertujuan untuk menilai pengalaman anda dalam pengujian perisian. Jawapan anda akan membantu kami memahami bagaimana pengujian perisian dilaksanakan dalam organisasi anda dan alat yang anda gunakan*

Does your organization currently implement software testing in the software development process? *

*Adakah organisasi anda kini melaksanakan ujian perisian dalam proses pembangunan perisian?*

- ◯ Yes/Ya
- ◯ No/Tidak

Have you heard of Bug Tracking Tool? *

*Pernahkah anda mendengar mengenai Bug Tracking Tool?*

- ◯ Yes/Ya
- ◯ No/Tidak

Are you currently using a Bug Tracking Tool in your team? If yes, what Bug Tracking Tool are you using? *

*Adakah anda sedang menggunakan Bug Tracking Tool dalam organisasi anda? Jika ya, apakah Alat/Bug Tracking Tool yang anda gunakan?*

- ◯ No/Tidak
- ◯ Other...

Do you think Software Testing is crucial to the success of a Software Development Project? *

*Adakah anda berasa bahawa aktiviti Pengujian Perisian adalah salah satu faktor kejayaan Projek Pembangunan Perisian?*

- ◯ Yes/Ya
- ◯ No/Tidak

Section 3 of 4

Part 2: Artificial Intelligence and Integration in Software Tools

This section focuses on assessing your familiarity with Artificial Intelligence (AI) and its applications within software tools. Your responses will help us determine how AI can potentially enhance your development practices and the use of intelligent systems in your projects.

*Bahagian ini fokus kepada pengetahuan anda mengenai Kecerdasan Buatan (AI) dan aplikasinya dalam alat perisian. Jawapan anda akan membantu kami menentukan bagaimana AI berpotensi untuk meningkatkan aktiviti pembangunan perisian anda dan penggunaan Kecerdasan Buatan (AI) dalam projek anda.*

Have you heard of Artificial Intelligence (AI)? *

*Adakah anda pernah mendengar tentang Kecerdasan Buatan (AI)?*

- ◯ Yes/Ya
- ◯ Yes, but I have never tried it/Ya, tetapi belum pernah mencubanya
- ◯ No/Tidak

Have you ever used Artificial Intelligence (AI) (ex. ChatGPT, Copilot etc) to complete your tasks? *

*Adakah anda pernah menggunakan Kecerdasan Buatan (contoh: ChatGPT, Copilot, dan sebagainya) untuk menyelesaikan tugas anda?*

- ◯ Yes/Ya
- ◯ Sometimes/Kadang-Kadang

Have you ever used Artificial Intelligence (AI) (ex. ChatGPT, Copilot etc) to complete your tasks? *

*Adakah anda pernah menggunakan Kecerdasan Buatan (contoh: ChatGPT, Copilot, dan sebagainya) untuk menyelesaikan tugas anda?*

- ◯ Yes/Ya
- ◯ Sometimes/Kadang-Kadang
- ◯ No/Tidak

Do you believe integrating AI into any Software Tools is beneficial? *

*Adakah anda percaya bahawa penglibatan Kecerdasan Buatan (AI) ke dalam mana-mana Alat Perisian/Software adalah bermanfaat?*

- ◯ Yes/Ya
- ◯ No/Tidak

What benefits do you expect from using AI during Software Development? *

*Apakah manfaat yang anda jangkakan daripada penggunaan Kecerdasan Buatan (AI) dalam Pembangunan Perisian?*

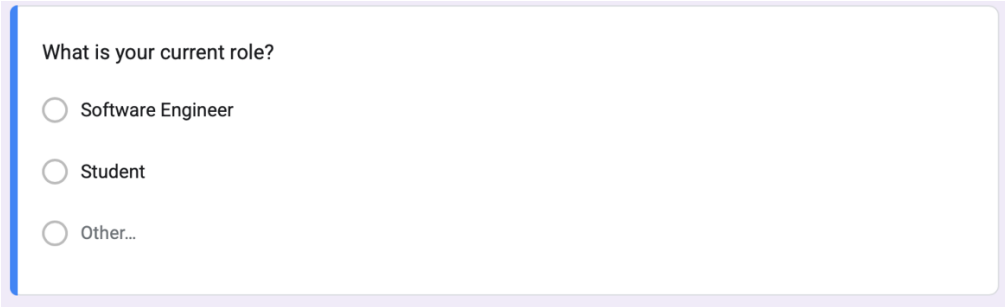
- ☐ Faster task completion/Penyelesaian tugas yang pantas
- ☐ Improved accuracy/Meningkatkan ketepatan dalam tugasan
- ☐ Reduced manual work/Mengurangkan tenaga kerja
- ☐ Better insights and analytics/Pengetahuan dan analitik yang lebih baik
- ☐ Other...

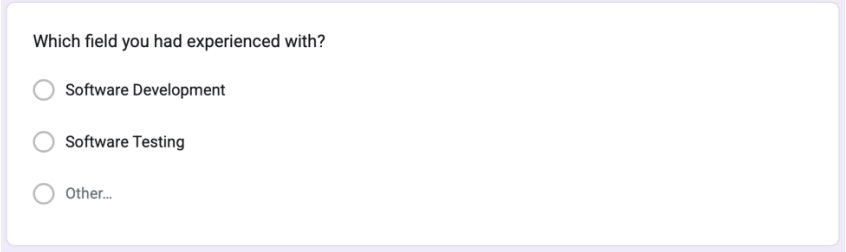
Do you have any concerns about integrating AI into Software Development? If so, please specify. *

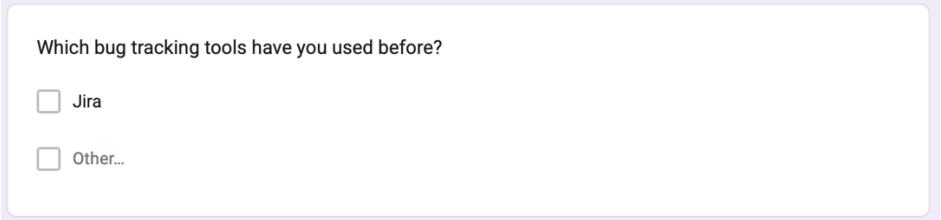
*Adakah anda mempunyai sebarang kebimbangan mengenai penglibatan Kecerdasan Buatan (AI) dalam Pembangunan Perisian? Jika ya, sila nyatakan.*

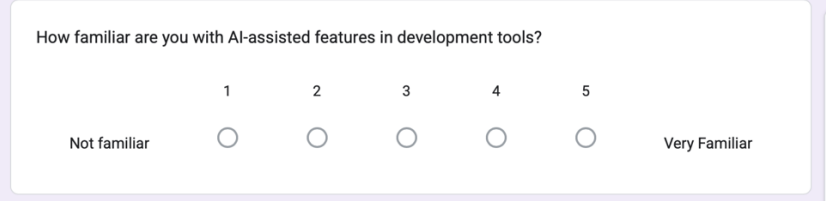
Long-answer text

# APPENDIX B: USABILITY TESTING QUESTIONS



**Team Management Module**

This section evaluates your experience using the team management module

How is your experience when creating and managing the team?

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Very Bad | ○ | ○ | ○ | ○ | ○ | Very Good |

Was anything confusing as you tried to manage the team?

Long-answer text

**Project Management Module**

This section evaluates your experience using the project module

How is your experience when creating and managing the project?

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Very Bad | ○ | ○ | ○ | ○ | ○ | Very Good |

Was anything confusing as you tried to manage the project?

Long-answer text

**Issue Management Module**

This section evaluates your experience using the issue management module

How is your experience when creating and managing the issue?

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Very Bad | ○ | ○ | ○ | ○ | ○ | Very Good |

Was anything confusing as you tried to manage the issue?

Long-answer text

**Project Summary**

This section evaluates your experience using the issue management module

How is your experience when viewing the project summary?

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Very Bad | ○ | ○ | ○ | ○ | ○ | Very Good |

Was anything confusing as you tried to view the project summary?

Long-answer text

**AI and Machine Learning Integration Module**

This section evaluates your experience using the Ai-features that exists in the system

Did the bug severity prediction align with your own expectations?

- ○ Yes
- ○ No
- ○ Maybe

Did the similar bug reported suggestions make sense to you?

- ○ Yes
- ○ No
- ○ Maybe

Was the bug summarization feature useful for the team?

- ○ Yes
- ○ No
- ○ Maybe

**Overall System**

This section aims to assess your overall experience with the system along with the Ai features. Your feedback will help identify strengths, usability concerns, and areas for improvement, ensuring the system continues to meet user needs effectively.

Please tick the best option for the following questions? *

| | Excellent | Good | Average | Poor |
|---|---|---|---|---|
| How do you rate th... | ○ | ○ | ○ | ○ |
| How do you rate gr... | ○ | ○ | ○ | ○ |
| How do you rate in... | ○ | ○ | ○ | ○ |
| How do you rate in... | ○ | ○ | ○ | ○ |

Would you rely on the AI for making decisions and analyzing issues? Why or why not?

Long-answer text

How does this compare to the tools you previously or currently use?

Long-answer text

# APPENDIX C: REPORT SIMILARITY PERCENTAGE